IMPROVEMENT OF ABS BRAKING ANALYSIS TOOL

GOODYEAR INNOVATION CENTER

Life Cycle Mechanics

EMEA - Luxembourg

15/02/2017 – 28/07/2017

Autor	*Stéphane Barroso*
Tutor	*Lassi Hartikainen*
Supervisor teacher	*Alain Oustalloup*

Contents

1 - Abstract

Within the last few decades, many efforts have been aimed at preventing car crash and reducing the victims number of traffic accident. One of the bigger innovation in this domain is the ABS, which prevent wheels of a vehicle to locking up and the vehicle from becoming uncontrollable.

Considering that tires are the most important part of a car and to provide the best product, tires companies like Goodyear works to know what is the effect of an ABS on a tire during braking. To do that, engineers developed a lot of tool to collect, to analyze and to interpret data. The aim of this project was to improve and regroup all ABS braking analysis tools into one software.

2 - Acknowledgements

I would like to express my deepest gratitude to all of those who made my internship possible.
First to the Goodyear Innovation Center Luxembourg which trust in me to complete this project and helped me to find an apartment. To my school the ENSEIRB-MATMECA part of the Institut Polytechnique at Bordeaux (INP Bordeaux).
To all my third-year teachers, who gave me the knowledge and mindset to apply for this internship.
To my tutor, Lassi Hartikainen who assisted and guided me during 6 months and all engineers from the Tire Vehicle Mechanics department who worked with me: Tony Deroo, Julien Majerus, Kanwar Bharat Singh, Stefano Picchiotti, Paul Joubert, Srikanth Sivaramakrishnan and Michael Selig who attended to my meeting and tested my program during this project, to Vasile Ciorna who help me to setup my computer, to our manager Edouard Michel and to our chief engineer Matt Kaufman.
To Alain Oustalloup who was my supervisor teacher during this internship.
Finally, I would like to thank my family and particularly my mother who helped me and supports me during all my studies, also I have a thought for my grandparents who always believed in me and motivated me to make greats things but are no longer here to see the result.

3 - Introduction

As part of my last year of engineering school at ENSEIRB-MATMECA (Ecole Nationale Supérieur Electronique, Informatique, Télécommunications, Mathématiques et Mécanique de Bordeaux). I needed to do an internship of 6 months abroad related to my last year option in AM2AS (Automation, Mechanics, Aeronautics, Automotive and Space).

I did my two previous internships respectively in a medium and a small company. That is why I aimed for a large one for this internship. This way I could experience every type of company during my three years of engineering school.

The Goodyear Tire & Rubber Company is an American multinational tire manufacturing company. Goodyear manufactures tires for automobiles, commercial trucks, light trucks, motorcycles, SUVs, race cars, airplanes, farm equipment and heavy earth-mover machinery. There are a lot of internship in this company, they hire around 30 interns each year just in Luxembourg.

The internship took place in the Life Cycle Mechanics group, part of Tire Vehicle Mechanics department (R&D department of Europe Middle East and Africa Goodyear Innovation Center Luxembourg).

I will first describe the company, then detail each part of the project and the execution of the project. And finally, I will discuss about the result.

4 - The Company

4.1 - Goodyear

The Goodyear Tire & Rubber Company is an American multinational tire manufacturing company founded in 1898 by Frank Seiberling. The Goodyear global headquarter is in Akron (OHIO, USA).

The Goodyear complex in Luxembourg where my internship took place consist of:

- Goodyear Innovation Center (GIC*L)
- Tire Plant
- Mold Plant
- Regional Calendar Center * Europe
- Luxembourg Mounting Center (LMC)
- EMEA Regional Offices

4.2 - Brands

Goodyear produce tires with several brands, the most important are:

- Goodyear
- Dunlop Tires
- Fulda
- The Kelly Springfield Tire Company
- Sava
- Debica

4.3 - Operations

Tire Engineering:

- Construction Development Consumer OE
- Construction Development Consumer Replacement
- Construction Development Heavy Tires
- Tread & Cavity Development
- Engineering Mechanics

Materials Science:

- Materials Technology
- Raw Materials & Compound Development
- Reinforcement Technology & Compound Development
- Metrology (Physical & Analytical Labs)

Tire Performance Prediction:

- Production Evaluation
- Tire-Vehicle Mechanics
- Simulation Technology

Innovation Prototyping Lab Luxembourg:

- Provides prototype, experimental / development and low volume commercial tires

4.4 - Revenue

Goodyear Net Sales or Revenues in 2016 15.16B USD*

4.5- Competition

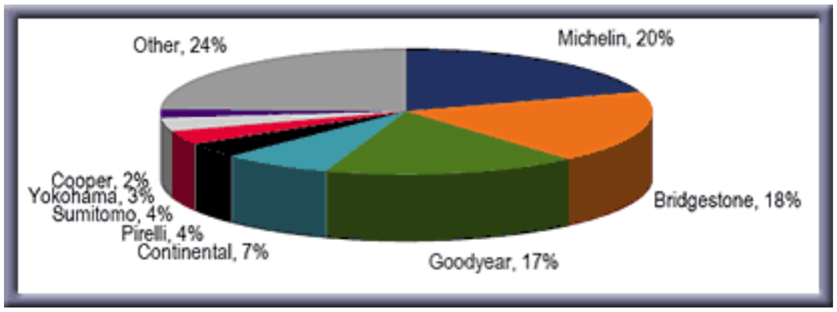


Figure : Global Market Share - Tire Manufacturer

There are 3 big company which lead 60% of the market in the tire industry, Michelin, Bridgestone and Goodyear. The leftover is divided between smaller companies.

4.6 - Other information

In Goodyear, a lot of measure are taken concerning the safety of the employee. This company wants to have their employees in the best condition to realize the best work possible.

Goodyear wants to provide the best growth possibilities for everyone who works in the company, many courses and training are organized to allow people to learn and discover new domains.

The security concerning IT has a big part in the company, every software or hardware needs to be approved by the manager before downloading/buying. And employees have the possibility to call an external service to solve their IT issues.

5 - Mission Objective

5.1 - Internship subject

The aim of this project was to group all the abs braking analysis tools into one and unique software which can be launched without running Matlab. Since number of Matlab/toolbox licenses are limited in the company, this software will permit engineers to work at any moment without taking the license from someone who might need it. Another part was to improve and complete currents scripts used by the analysis tool and to make them faster and more efficient.

5.2 - Specification

- Building a software with a graphical user interface
- The software should regroup the following files:

- A Preprocessing script and configuration file for the vehicle
- A Diagnostic script and configuration file used in case there is an error during processing
- A Processing script and configuration file
- A Reporting script and configuration file for plotting the data and generating tables
- A Plotting script and configuration file for another data source
- A script that will launch the Reporting and the Plotting script

- The software should be able to run independently from Matlab
- It should be easy to modify or maintain
- The plotted figure should also display help or information to inform users on how to interpret the results.
- Upgrade for existing scripts (speed, new functionalities)

5.3 - Project approval

Meetings with associates from Luxembourg, Hanau (Germany) and Akron (United states) were organized to be sure that the software was in line with the need of each user. The meetings allowed to collect feedback about bugs or about potential changes that needed to be made.

Each week a meeting with my supervisor Lassi Hartikainen was organized to discuss the advancement of the project and to set priorities for the next week.

Every function or script that I made was validated by my tutor and was committed to the server to be used by all users. This way, the feedback was instant, which allowed me to be reactive and to do modifications if needed.

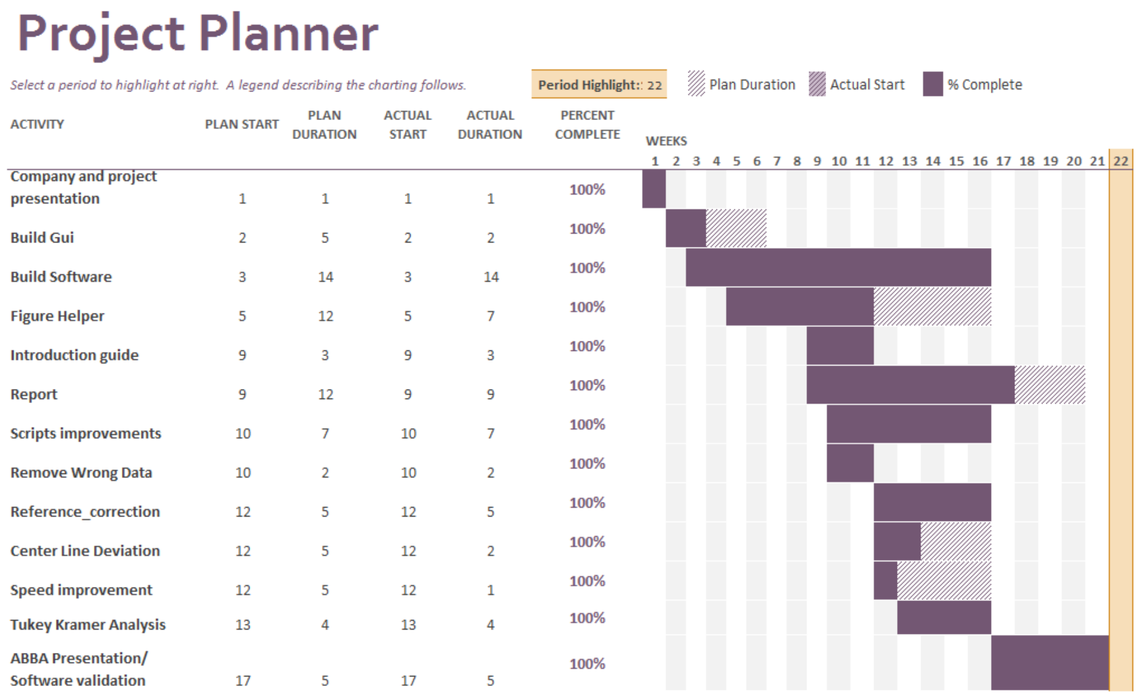
5.4 - Means and resources

During this internship, I got a desk, a phone, a laptop, an external keyboard, an external screen and an access to office supplies such as paper, pencil, notebook etc.

To be able to work without having to wait for a shares license from the server, my computer was setup with its own Matlab license, while many toolboxes were available on the server. I had a full access to internet and to some shared drives, as well as software from Microsoft: Word, Excel, PowerPoint, Skype and Outlook. I was also free to organize meetings when I needed to discuss with someone about my project.

5.5 - Scheduling

The figure below shows how the project was completed



Project Planner

Select a period to highlight at right. A legend describing the charting follows.

Period Highlight:: 22 ▨ Plan Duration ▨ Actual Start ■ % Complete

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Company and project presentation	1	1	1	1	100%
Build Gui	2	5	2	2	100%
Build Software	3	14	3	14	100%
Figure Helper	5	12	5	7	100%
Introduction guide	9	3	9	3	100%
Report	9	12	9	9	100%
Scripts improvements	10	7	10	7	100%
Remove Wrong Data	10	2	10	2	100%
Reference_correction	12	5	12	5	100%
Center Line Deviation	12	5	12	2	100%
Speed improvement	12	5	12	1	100%
Tukey Kramer Analysis	13	4	13	4	100%
ABBA Presentation/ Software validation	17	5	17	5	100%

Figure 2: Project schedule

Some important dates:

16/03/2017(Week4)	First presentation of the software to associates
21/04/2017(Week10)	Second presentation of the software to associates
19/07/2017(Week21)	ABBA Presenting/Training to engineer
25/07/2017(Week22)	Presentation of the project to the TVM department

The first step was the design and the building a GUI in Matlab environment. Then after the first meeting and the validation of the interface, a help function for figures that are displayed was to be created. A layout for the interface and how to use the figure helper was written.

The improvement of scripts became the priority after the development of the interface and the helper. Week after week, functions were added to the code to complete and improve it.

The last part was a technical report for the company and presentation to explain the full content of the ABS braking analysis software (this project and all tools which were developed before). During the last month, the newly developed functions were validated.

6 - Project Completion

6.1 - Analysis of the specifications and overall design

6.1.1 - Regrouping scripts

An interface needed to be built to be able to start each script. Matlab allows creation of GUIs and executables. A new interface designer was implement in Matlab 2016: AppDesigner, so I decided to use this one for the project. Particularly, the new class matlab.apps.AppBase related to AppDesigner can now be used without the AppDesigner, which allows the creation of an interface from a script.

Basically, each individual script works like this:

- Configuring the related ini-file with parameters needed
- Running the script on Matlab
- Selecting the data
- Displaying and saving the results or/and store.

Since each file works the same way, and to not have to access the ini-file each time a parameter needs to be changed, the current configuration files needed to be kept. A related GUI which is flexible and easily manageable needed to be built.

To display some help or information on the figure, developer shouldn't have to change the code to add information and/or help. Also, there are already some files to explain and how some of the script work or how to interpret the figures. To not have to rewrite what already exists, the file should be opened directly from a figure.

To accomplish this, several functions needed to be develop:

- ini_to_struct.m: reads an ini-file and returns a structure
- struct_to_ini.m: writes a structure into an ini-file
- ABBA.m: script of the interface and the main file of the project
- Dhfclass.m: controls to help windows on the figures

6.1.2 - Script improvements

The execution time needed to be improved and new functions needed to be added. To avoid the need for external software, all functionalities were implemented into Matlab.

The list of the new/improved functions is below:

- vboread.m: reads the raw data files
- ref_correction.m: corrects for reference evolution in results (before done in external software)
- remove_wrong_data.m: removes peaks/dropouts from raw data
- tukey_kramer.m: compares statistical significance between results (before done in external software)

6.2 - Detailed design

6.2.1 - Interface and software building

To start avoid having to open the configuration file each time the user wants to change a parameter, an interface to manage the parameters and to run the scripts was needed. The interface should run all the m-files that are used for braking analysis, using the same parameters as the configuration files.

6.2.1.1 - Meeting results

The interface which was presented on the first meeting was accepted. The code used current configuration files and Matlab files. In addition, the interface was generated based on the configuration files, which allowed it to be very flexible. We also discussed that some information should be added to figures that are ploted by the different tools. Some analysis steps that were made using external software needed to be implemented in Matlab:

- The reference correction
- The tool to remove peaks/dropouts (measure error during acquisition) in a set of data
- The Tukey Kramer analysis (statistical analysis to compare multiple sets of data)

6.2.1.2 - Software Building

Since all the scripts were already running on Matlab, we chose to use Matlab to build the user interface. There are several ways to design an interface in Matlab:

- with a normal script where elements are inserted manually.

- with Guide which is an earlier tool to design a GUI
- with App Designer which is the latest design tool implemented by Mathworks

6.2.1.2.1 - Matlab App Designer

App Designer is a new interface builder which is available in Matlab since the version 2016. This is the easiest way to design an app on Matlab. But there are some issues when the script is too long and you cannot edit each part of the script easily. Therefore the m file version of the mlapp file generated by App Designer was used. This can be done by using the function ***mlapp2classdef [4]***.
The m-file can be run as a normal script and the most important the code is flexible. It is easier to proceed like this because the App Desgner tool on Matlab is not convenient to use for an advanced software. Also, this allows to work on the interface with a text editor and not necessarily with Matlab.

6.2.1.2.2 - Keeping original files

The scripts and configuration files are on a server and are used by many users. The decision was made to design an app that can works with files that already exist.

The interface reads the existing configuration files to create the corresponding parameters in the GUI, which makes it easy to manage without changing the code.

To create the figure helper, a class of functions was created. These functions read all currents figures that are displayed, can change the properties and store information into a matfile. This way users can add and remove information shown in the figures without adding code and without having to recompile the software.

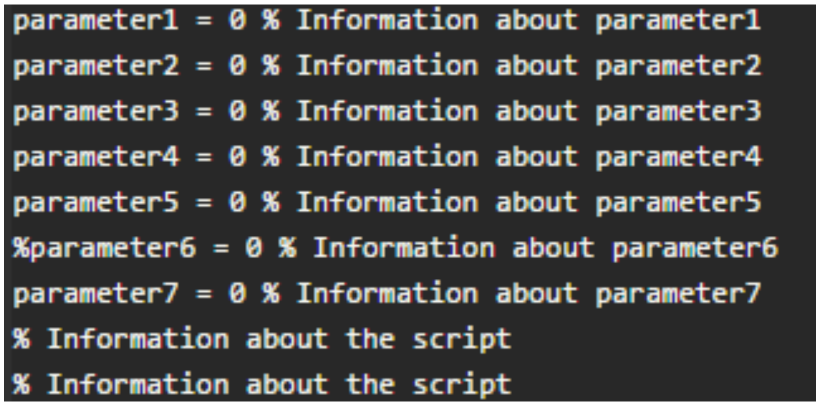
Finally, each improvement was implemented in a function, so that they can be used for others projects too.

6.2.1.3 - Code description

In this part, only parts of code will be shown, nevertheless full code can be found in the annex.

6.2.1.3.1 - Ini_to_struct

This function takes an ini file as input and returns a structure.

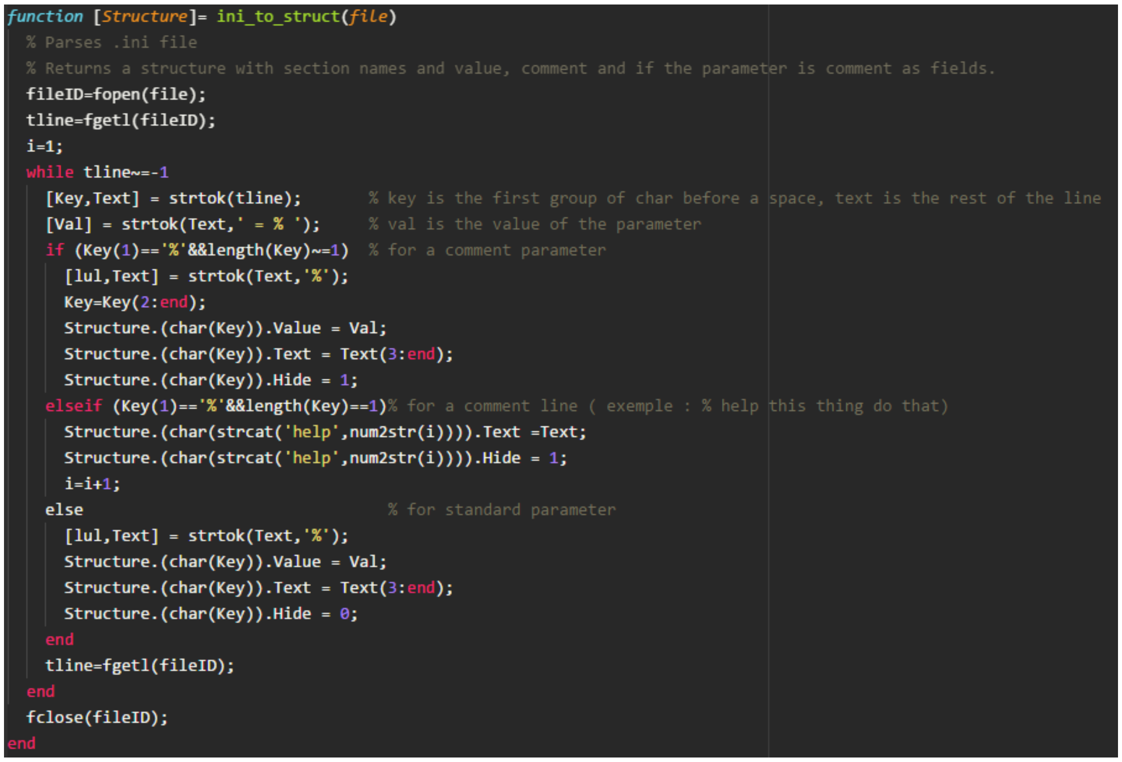


```
parameter1 = 0 % Information about parameter1
parameter2 = 0 % Information about parameter2
parameter3 = 0 % Information about parameter3
parameter4 = 0 % Information about parameter4
parameter5 = 0 % Information about parameter5
%parameter6 = 0 % Information about parameter6
parameter7 = 0 % Information about parameter7
% Information about the script
% Information about the script
```

Figure 3: Example of a configuration file

Every script has its own configuration file, but they are all in the same format. There are three types of information that can be on the configuration file:

- A commented line
- A parameter with a value and a related information
- A commented parameter with a value and a related information

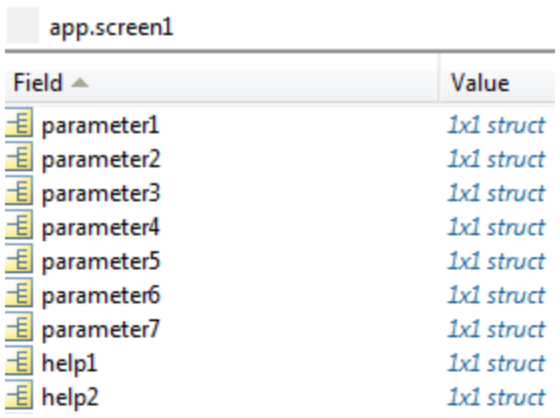


```
function [Structure]= ini_to_struct(file)
  % Parses .ini file
  % Returns a structure with section names and value, comment and if the parameter is comment as fields.
  fileID=fopen(file);
  tline=fgetl(fileID);
  i=1;
  while tline~=-1
    [Key,Text] = strtok(tline);       % key is the first group of char before a space, text is the rest of the line
    [Val] = strtok(Text,' = % ');     % val is the value of the parameter
    if (Key(1)=='%'&&length(Key)~=1)  % for a comment parameter
      [lul,Text] = strtok(Text,'%');
      Key=Key(2:end);
      Structure.(char(Key)).Value = Val;
      Structure.(char(Key)).Text = Text(3:end);
      Structure.(char(Key)).Hide = 1;
    elseif (Key(1)=='%'&&length(Key)==1)% for a comment line ( exemple : % help this thing do that)
      Structure.(char(strcat('help',num2str(i)))).Text =Text;
      Structure.(char(strcat('help',num2str(i)))).Hide = 1;
      i=i+1;
    else                                % for standard parameter
      [lul,Text] = strtok(Text,'%');
      Structure.(char(Key)).Value = Val;
      Structure.(char(Key)).Text = Text(3:end);
      Structure.(char(Key)).Hide = 0;
    end
    tline=fgetl(fileID);
  end
  fclose(fileID);
end
```

Figure 4: ini_to_struct.m

The figure 4 can be explained as follows:

- the text is read line by line and converted into structure based on the format of the line.



app.screen1

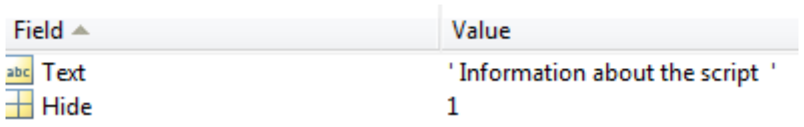
Field	Value
parameter1	1x1 struct
parameter2	1x1 struct
parameter3	1x1 struct
parameter4	1x1 struct
parameter5	1x1 struct
parameter6	1x1 struct
parameter7	1x1 struct
help1	1x1 struct
help2	1x1 struct

Figure 5: Example of structure generated with ini_to_struct

Figure 5 shows the output from the structure related to the ini file shown in figure3. In each sub structure, two or three fields can be found (figure6 and figure7):

- Value: The value of the parameter. (Char)

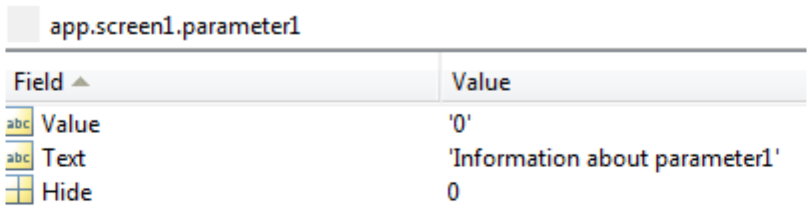
- Text: Information related to the parameter or to the script. (Char)
- Hide: The information if the parameter/information is commented or not. (Bool)



Field	Value
Text	' Information about the script '
Hide	1

Figure 6: Example struct for a commented line (help1)

- A commented line will be stored in a sub structure called "*help(i).Text*" with the field *Hide* set to 1.

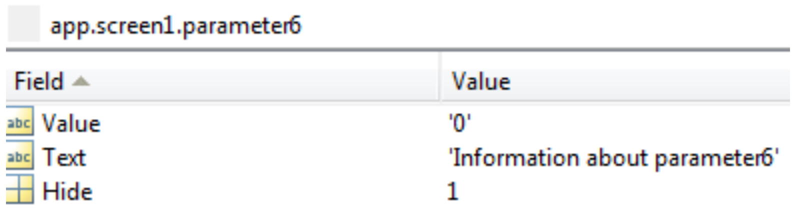


app.screen1.parameter1

Field	Value
Value	'0'
Text	'Information about parameter1'
Hide	0

Figure 7: Example struct for a parameter (parameter1)

- A parameter will be stored in the field *Value*, the information on the field *Text* and the field *Hide* set to 0.



app.screen1.parameter6

Field	Value
Value	'0'
Text	'Information about parameter6'
Hide	1

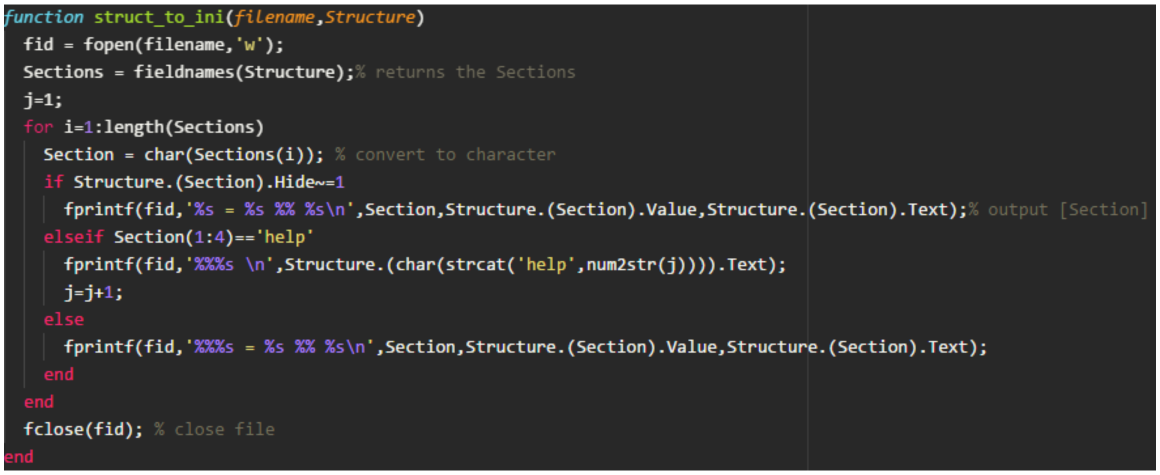
Figure 8: Example struct for a hidden parameter (parameter6)

- A commented parameter will be stored in the field *Value*, the information on the field *Text* and the field *Hide* set to 1.

6.2.1.3.2 - Struct_to_ini

This function takes struct and a name of file as input and returns an ini file. The information contained in the struct can have two forms:

- A char field named *Value*, a char field *Text* and a Boolean field *Hide*. (Figure 7/8)
- A char field *Text* and a Boolean field *Hide*. (Figure 6)



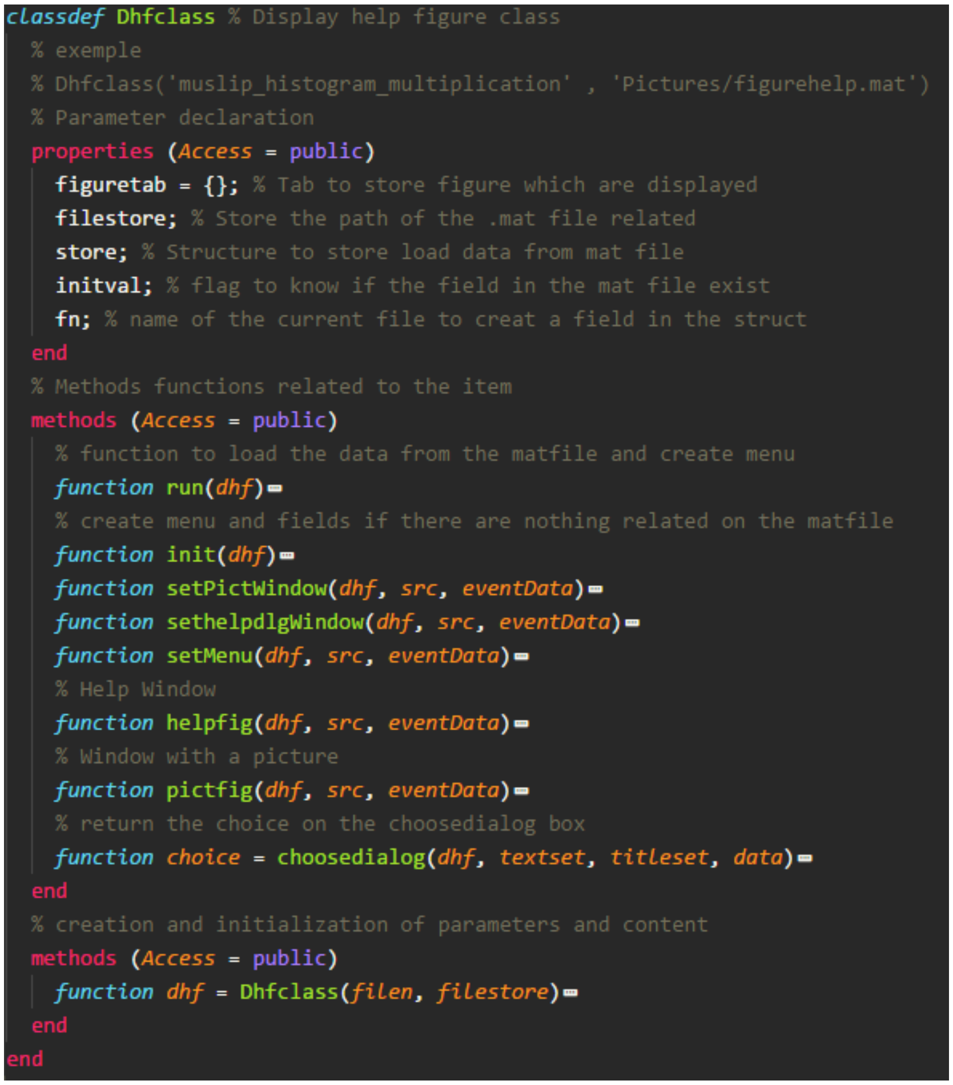
```
function struct_to_ini(filename,Structure)
  fid = fopen(filename,'w');
  Sections = fieldnames(Structure);% returns the Sections
  j=1;
  for i=1:length(Sections)
    Section = char(Sections(i)); % convert to character
    if Structure.(Section).Hide~=1
      fprintf(fid,'%s = %s %% %s\n',Section,Structure.(Section).Value,Structure.(Section).Text);% output [Section]
    elseif Section(1:4)=='help'
      fprintf(fid,'%%%s \n',Structure.(char(strcat('help',num2str(j)))).Text);
      j=j+1;
    else
      fprintf(fid,'%%%s = %s %% %s\n',Section,Structure.(Section).Value,Structure.(Section).Text);
    end
  end
  fclose(fid); % close file
end
```

Figure 9: struct_to_ini.m

It is the exact opposite of the ini_to_struct function presented before. It writes information that are contained in the structure (Figure 5) into the configuration file (Figure 4).

6.2.1.3.3 - Dhfclass

This class allows the user to display additional information on a figure using *Help items.*



```
classdef Dhfclass % Display help figure class
  % exemple
  % Dhfclass('muslip_histogram_multiplication' , 'Pictures/figurehelp.mat')
  % Parameter declaration
  properties (Access = public)
    figuretab = {}; % Tab to store figure which are displayed
    filestore; % Store the path of the .mat file related
    store; % Structure to store load data from mat file
    initval; % flag to know if the field in the mat file exist
    fn; % name of the current file to creat a field in the struct
  end
  % Methods functions related to the item
  methods (Access = public)
    % function to load the data from the matfile and create menu
    function run(dhf)
    % create menu and fields if there are nothing related on the matfile
    function init(dhf)
    function setPictWindow(dhf, src, eventData)
    function sethelpdlgWindow(dhf, src, eventData)
    function setMenu(dhf, src, eventData)
    % Help Window
    function helpfig(dhf, src, eventData)
    % Window with a picture
    function pictfig(dhf, src, eventData)
    % return the choice on the choosedialog box
    function choice = choosedialog(dhf, textset, titleset, data)
  end
  % creation and initialization of parameters and content
  methods (Access = public)
    function dhf = Dhfclass(filen, filestore)
  end
end
```

Figure 10: Dhfclass.m

The properties section lists different variables used in the code. The first method section lists the functions to access and modify the current figure. The second method section is used to initialize the class.

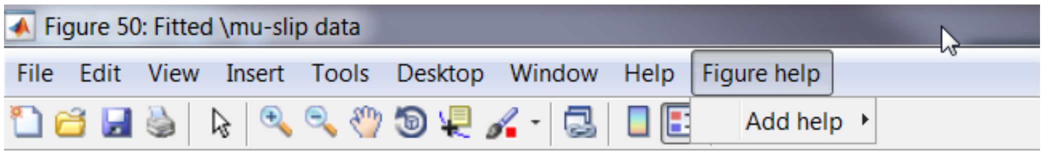


Figure 11: New menu

When the class *Dhfclass* is created, the toolbar from figure 11 with the new item "*Figure help*" will be added on all open figures.

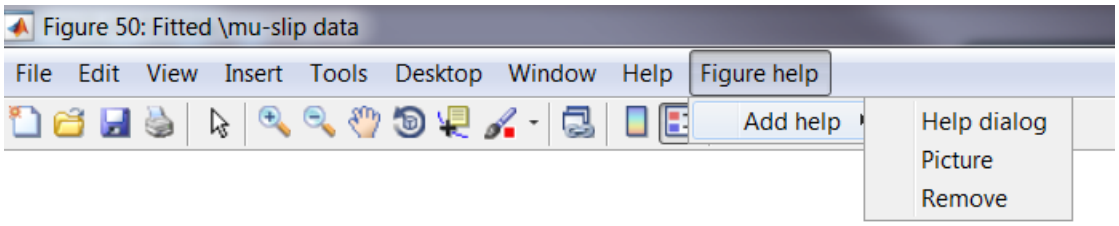
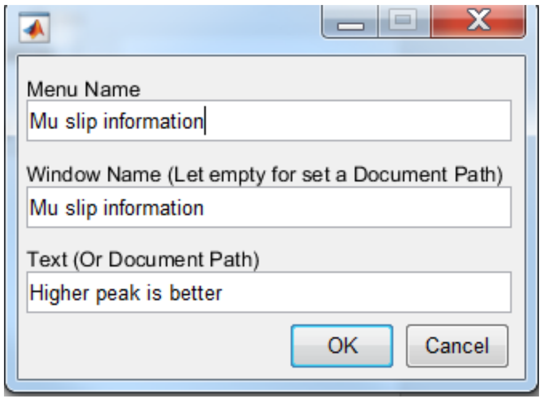


Figure 12: Menu content

Two types of help are available when the user clicks on the figure help button:

- a picture
- an help dialog window

6.2.1.3.3.1 - Functionality: Help dialog



Menu Name

Mu slip information

Window Name (Let empty for set a Document Path)

Mu slip information

Text (Or Document Path)

Higher peak is better

Figure 13: Help Text menu

If the help dialog option is chosen, it will open a new window where the user can select the name of the window and the text that he wants to add.

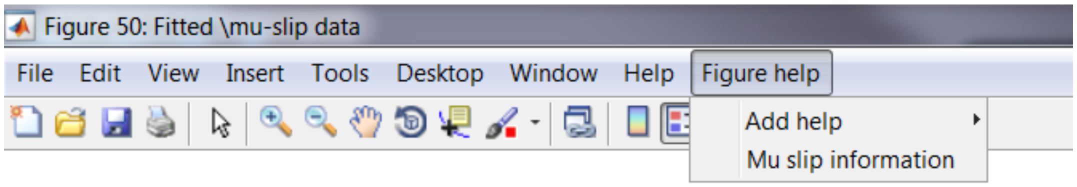
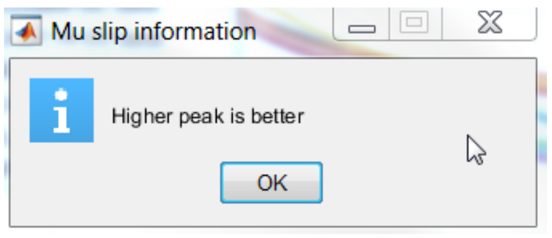


Figure 14: New help items

If a *Help item* is created, it will be available on the *Figure help* menu, allowing the user to click on it to display the information.


Mu slip information

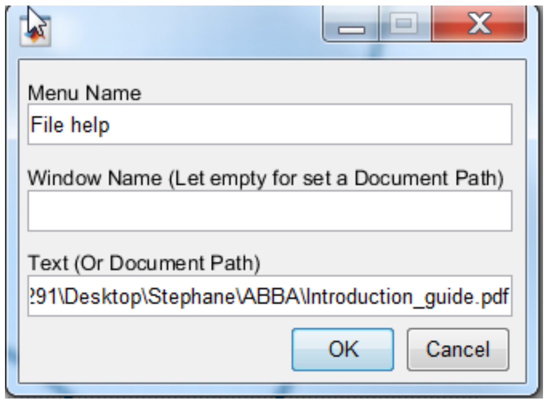
Higher peak is better

Figure 15: Example of Help window

Once the help dialog is displayed, it will contain the information given by the user.

6.2.1.3.3.2 - Functionality: Help document

We can also include a link to an external document or a folder. The menu below is displayed after clicking on the *Help dialog* button (figure12).


Menu Name
File help

Window Name (Let empty for set a Document Path)

Text (Or Document Path)
?91\Desktop\Stephane\ABBA\Introduction_guide.pdf

Figure 16: Document help menu

To open a file from the *Help items*, the path to the file should be written in the last line and the window name should be empty. This will allow the user to open an external document (pdf, word excel etc…) or a folder.

6.2.1.3.3.3 - Functionality: Help picture

The menu below is displayed by clicking on the Picture button (figure12).

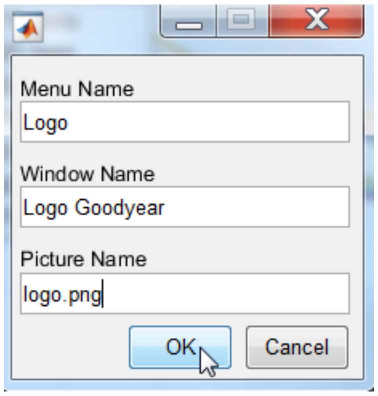


Figure 17: Picture menu

To open a picture from the menu, the name of the picture should be written in the last line and the picture should be store in the folder *Pictures*.

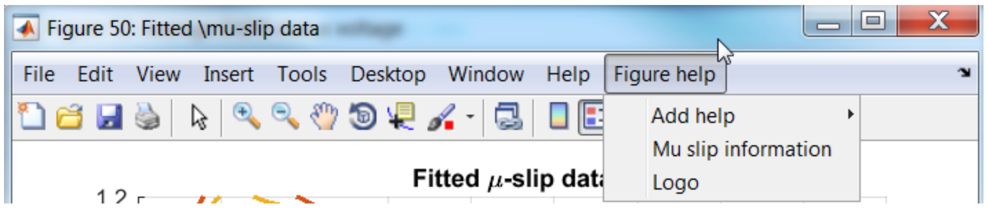


Figure 18: Picture item

The corresponding picture *Help items* is now added in the last line of the figure menu and can be selected to show the picture.

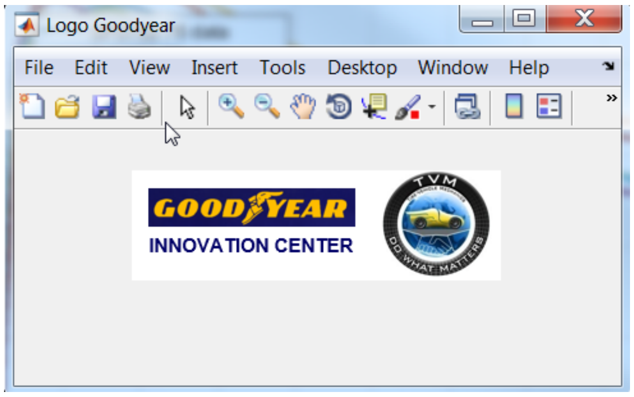


Figure 19: Image display

By clicking on the new item, a window containing the picture will be displayed.

6.2.1.3.3.4 - Functionality: remove helper

All *Help items* added can also be removed by using the *Remove helper* functionality.

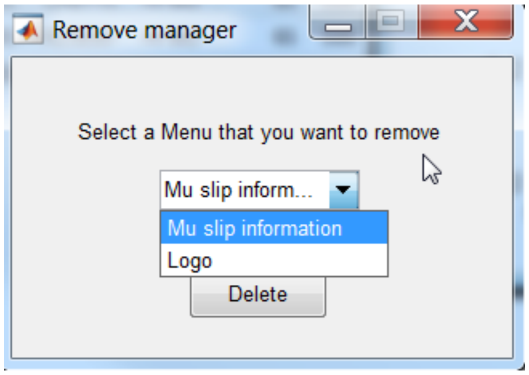


Figure 20: Remove manager

With the remove manager, and after selecting the *Help items*, the corresponding menu and item information will be removed.

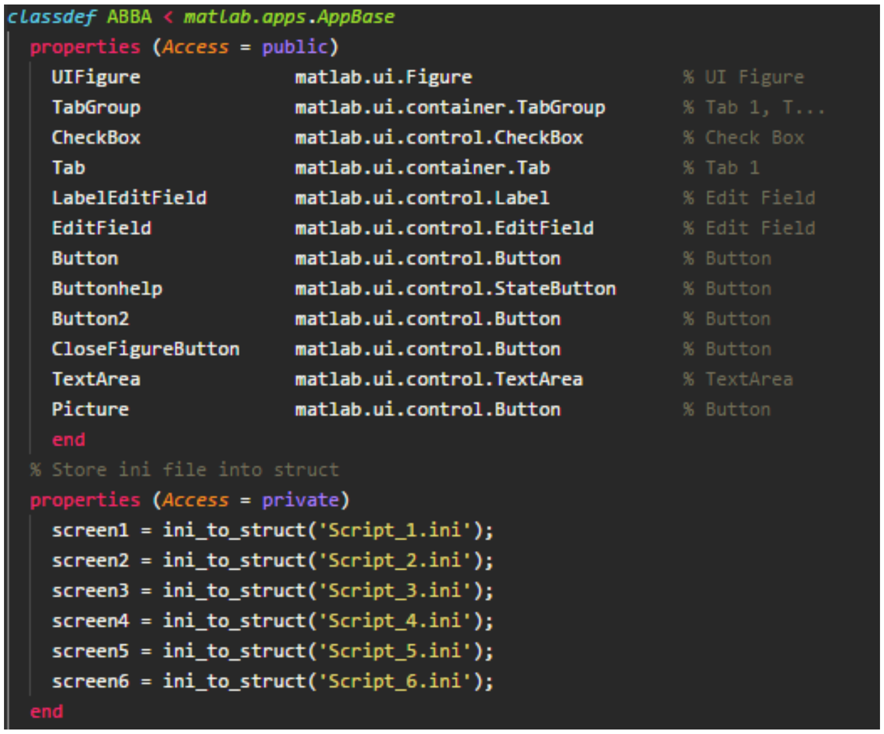
6.2.1.3.3.5 - Functionality: Saving information

All menus are stored in a *.mat*-file with the information about the name of the script that called the *Dhfclass*, the number of the figure and the type of the menu. This way, on the next time the script is run the new *Help items* will be displayed.

6.2.1.3.4 - ABBA

ABBA is the biggest part of the project. It is the interface which links each configuration file and allows to start each script. The class *ABBA* includes two properties sections and three methods sections.

6.2.1.3.4.1 - Properties



```
classdef ABBA < matlab.apps.AppBase
  properties (Access = public)
    UIFigure              matlab.ui.Figure                  % UI Figure
    TabGroup              matlab.ui.container.TabGroup      % Tab 1, T...
    CheckBox              matlab.ui.control.CheckBox        % Check Box
    Tab                   matlab.ui.container.Tab           % Tab 1
    LabelEditField        matlab.ui.control.Label           % Edit Field
    EditField             matlab.ui.control.EditField       % Edit Field
    Button                matlab.ui.control.Button          % Button
    Buttonhelp            matlab.ui.control.StateButton     % Button
    Button2               matlab.ui.control.Button          % Button
    CloseFigureButton     matlab.ui.control.Button          % Button
    TextArea              matlab.ui.control.TextArea        % TextArea
    Picture               matlab.ui.control.Button          % Button
    end
  % Store ini file into struct
  properties (Access = private)
    screen1 = ini_to_struct('Script_1.ini');
    screen2 = ini_to_struct('Script_2.ini');
    screen3 = ini_to_struct('Script_3.ini');
    screen4 = ini_to_struct('Script_4.ini');
    screen5 = ini_to_struct('Script_5.ini');
    screen6 = ini_to_struct('Script_6.ini');
  end
```

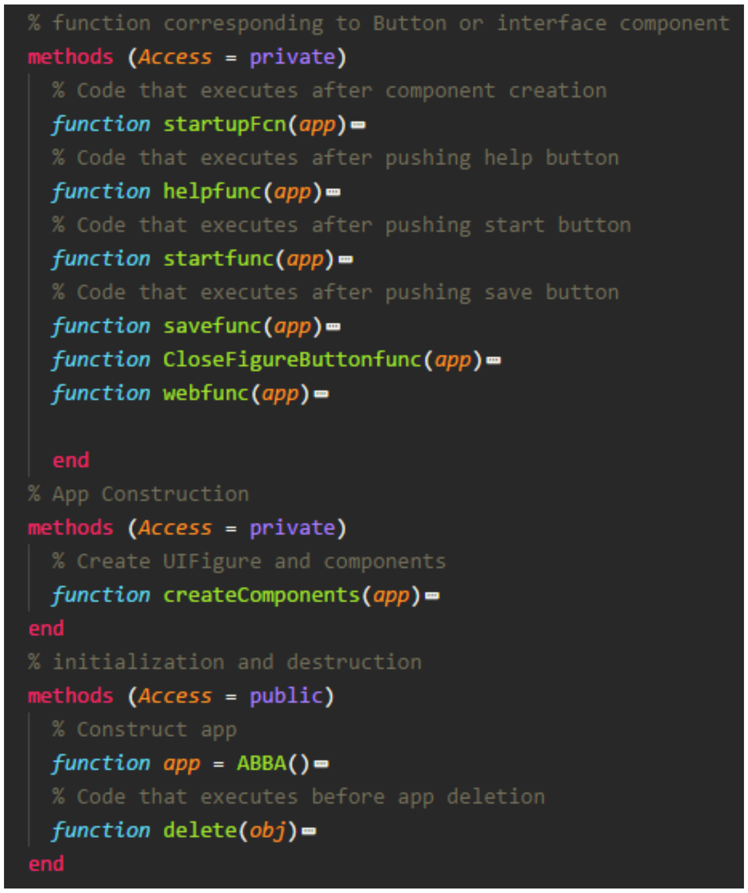
Figure 21: ABBA properties

The first properties sections include all elements that are on the GUI:

- *UIFigure* is the window
- *TabGroup* is the tab which contains the name of each script
- *Tab* corresponds to each element on the tab group

The second properties section includes the generation of six structs corresponding to six configuration files, which will be used to generate different items shown on the GUI.

6.2.1.3.4.2 - Methods



```
% function corresponding to Button or interface component
methods (Access = private)
  % Code that executes after component creation
  function startupFcn(app)
  % Code that executes after pushing help button
  function helpfunc(app)
  % Code that executes after pushing start button
  function startfunc(app)
  % Code that executes after pushing save button
  function savefunc(app)
  function CloseFigureButtonfunc(app)
  function webfunc(app)

  end
% App Construction
methods (Access = private)
  % Create UIFigure and components
  function createComponents(app)
end
% initialization and destruction
methods (Access = public)
  % Construct app
  function app = ABBA()
  % Code that executes before app deletion
  function delete(obj)
end
```

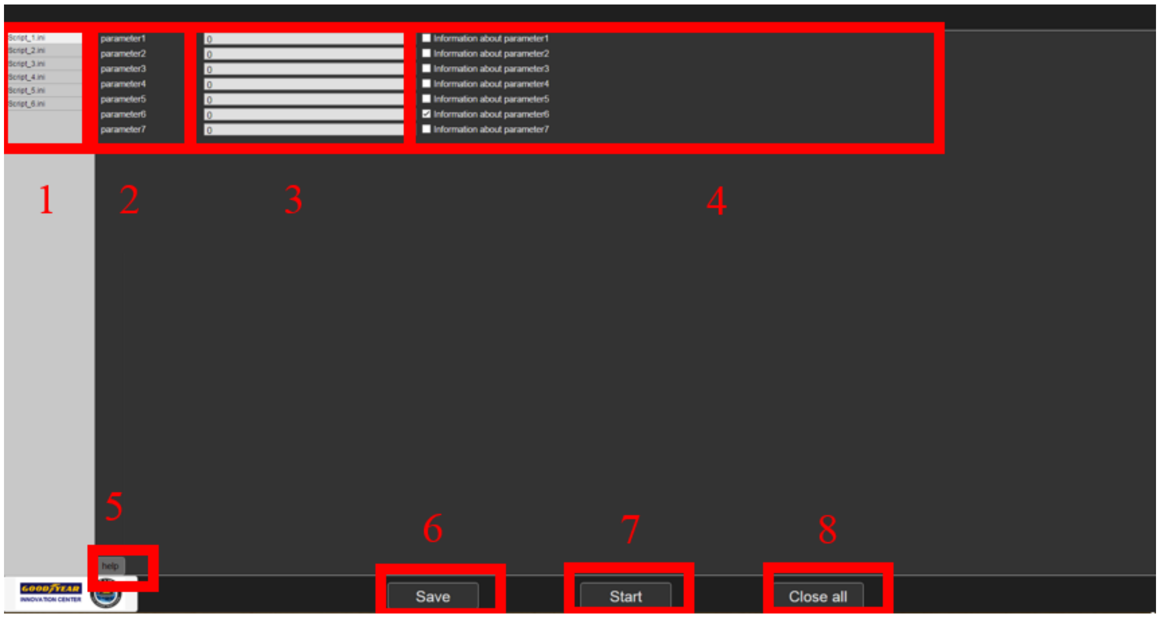
Figure 22: Abba Methods

On the first method section, there are five functions corresponding to the different buttons on the interface (callback). On the second method section, there is only one function which creates all components on the interface. The last method section includes functions for creating and deleting the application.

The app allows the user to change the parameters that will be used by the analysis script. Each parameter is read by the *ini_to_struct* function (figure21) and the interface is generated with the function *createComponents* in the second method section of ABBA (Annex). This way, if the analysis tool needs to be modified, the developer just needs to edit the script and the configuration file used for the analysis. The interface app will adapt to it. If the parameters are modified from the GUI, there is the *savefunc* function (the startupFcn function also calls *savefunc*) that uses the *struct_to_ini* function that will write the new information in the ini file.

6.2.1.3.4.1 – Interface presentation

The software interface generated by the code contains many elements as shown in the figure (23).



parameter1	0	☐ Information about parameter1
parameter2	0	☐ Information about parameter2
parameter3	0	☐ Information about parameter3
parameter4	0	☐ Information about parameter4
parameter5	0	☐ Information about parameter5
parameter6	0	✓ Information about parameter6
parameter7	0	☐ Information about parameter7

Figure 23: ABBA Graphical User Interface

Details of components that are on the GUI:

1- Tab table where the script can be selected

2- Parameters related to the selected script

3- Value of each parameter

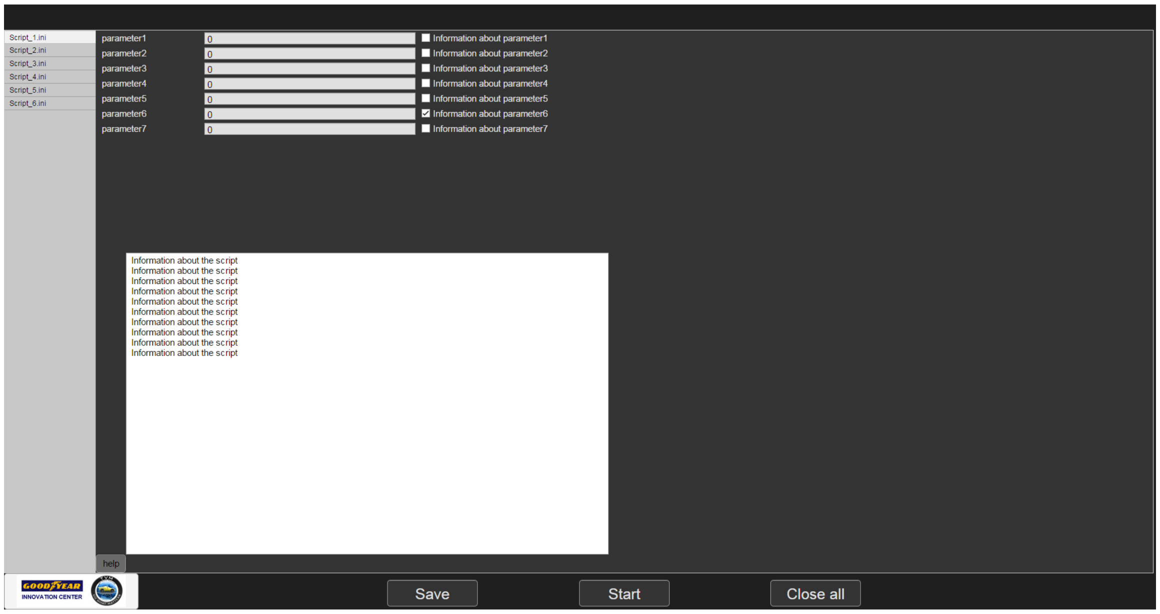
4- Information related to the parameters. The check boxes allow to comment out parameters in the ini files.

5- Help button that displays (figure 24) each text line that is commented out in the corresponding ini file.

6- Saves the current parameters values in the corresponding ini file

7- Starts the script related to the selected tab

8- This button will close all figures (equivalent to the "*close all*" in Matlab).



Script_1.ini	parameter1	0	☐ Information about parameter1
Script_2.ini	parameter2	0	☐ Information about parameter2
Script_3.ini	parameter3	0	☐ Information about parameter3
Script_4.ini	parameter4	0	☐ Information about parameter4
Script_5.ini	parameter5	0	☐ Information about parameter5
Script_6.ini	parameter6	0	✓ Information about parameter6
	parameter7	0	☐ Information about parameter7

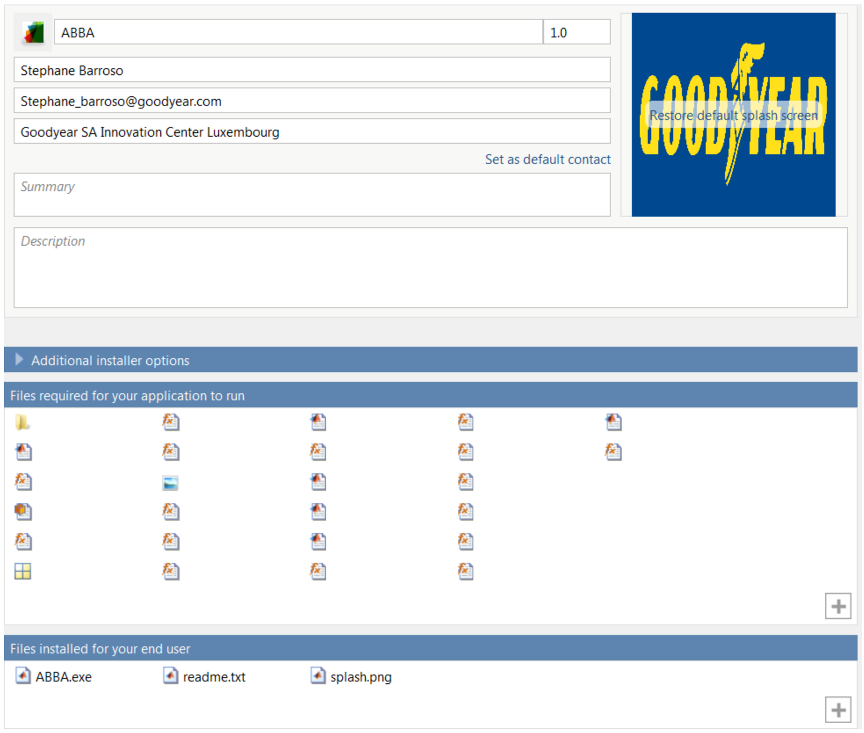
```
Information about the script
Information about the script
Information about the script
Information about the script
Information about the script
Information about the script
Information about the script
Information about the script
Information about the script
Information about the script
```

Figure 24: GUI Help Window

By adding a new line inside a configuration file, a parameter/information will be added to the corresponding tab. As explained before the interface is directly linked to all configuration files. The interface can also be used for others projects.

6.2.1.3.5 - Building the compiled version

With the *deploytool* command on Matlab, the m-file can be compiled and built into an executable.



ABBA 1.0

Stephane Barroso

Stephane_barroso@goodyear.com

Goodyear SA Innovation Center Luxembourg

Set as default contact
Summary
Description
Restore default splash screen
Additional installer options
Files required for your application to run
Files installed for your end user

ABBA.exe readme.txt splash.png

Figure 25: Matlab deploytool

Once all required files are linked in the compiler (Figure 25), an executable can be built.

Advantages of an executable:

- No need of Matlab
- No need of any Matlab toolboxes
- No need for Matlab/Toolbox licenses

Disadvantages:

- Script can no longer be modified without recompiling
- Figures that are displayed do not include all functions that are available when running in Matlab

6.2.2 Improvements of analysis script.

Another part of the project was the implementation of new functions to improve the efficiency of the analysis tools. As explained in the part 6.1.2, fore new functions were implemented:

- vboread.m: Reads a *.vbo*-file which contains data acquired during an instrumented vehicle test
- ref_correction.m: Normalizes data based on reference/control measurements (previously done with *Excel*)
- remove_wrong_data.m: Removes dropouts and peak in a set of data
- tukey_kramer.m: Performs a statistical Tukey Kramer analysis on a set of data (previously done with a statistical software)

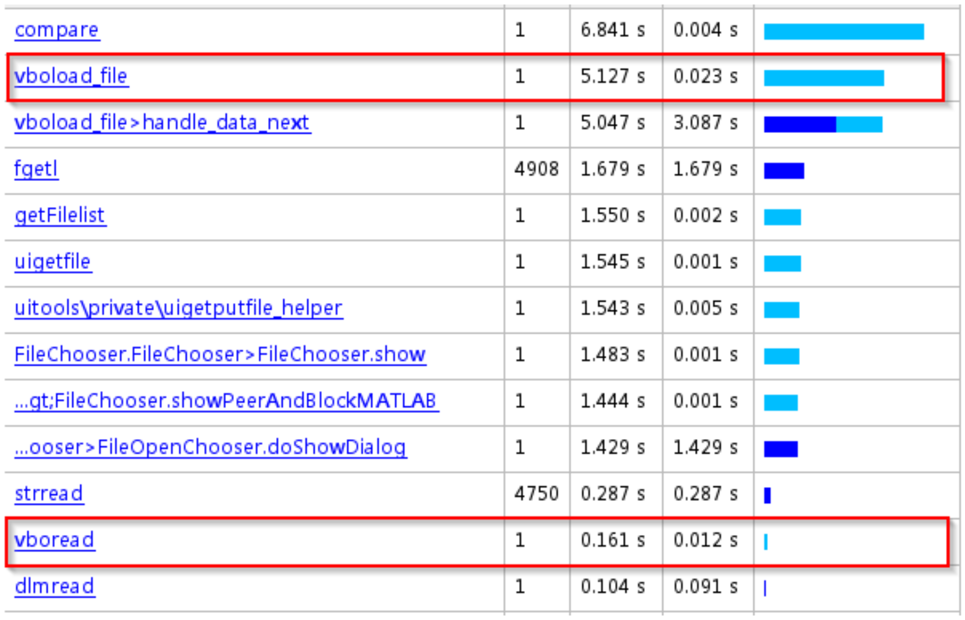
Note: The code can be found in the Annex.

6.2.2.1 vboread

The issues with some of the analysis scripts was the time that was needed to read data from files.

The figure below show that the original script used for loading data (***vboload_file [6]***) takes around 5 seconds to read the data.
In the example below the size of the numeric table is 4750 x 54.

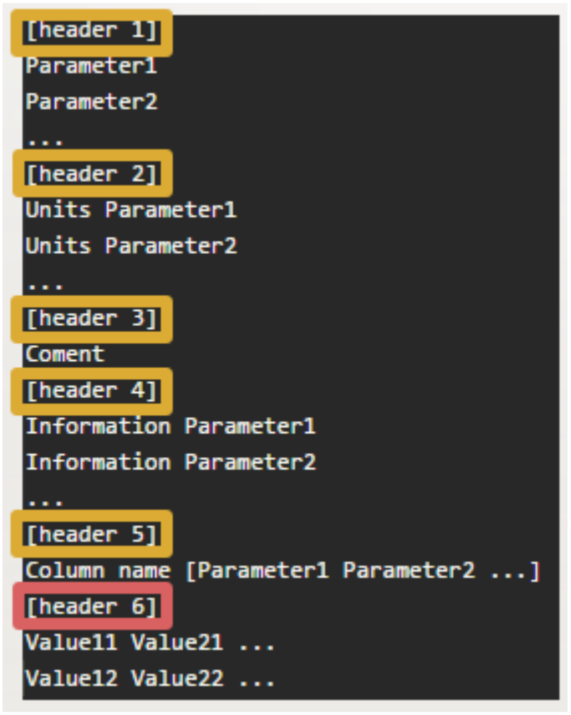
The original function read line by line the file and then collected all the information into a struct. But the longest part was to transform the data into a table of numeric values that can be read by Matlab.



compare	1	6.841 s	0.004 s	
vboload_file	1	5.127 s	0.023 s	
vboload_file>handle_data_next	1	5.047 s	3.087 s	
fgetl	4908	1.679 s	1.679 s	
getFilelist	1	1.550 s	0.002 s	
uigetfile	1	1.545 s	0.001 s	
uitools\private\uigetputfile_helper	1	1.543 s	0.005 s	
FileChooser.FileChooser>FileChooser.show	1	1.483 s	0.001 s	
...gt;FileChooser.showPeerAndBlockMATLAB	1	1.444 s	0.001 s	
...ooser>FileOpenChooser.doShowDialog	1	1.429 s	1.429 s	
strread	4750	0.287 s	0.287 s	
vboread	1	0.161 s	0.012 s	
dlmread	1	0.104 s	0.091 s	

Figure 26: Speed comparison

To solve this execution time problem, a new function was created: '*vboread.m'* which has the same input and output as ***vboload_file [6]***. But to increase the speed, the way that the file is read was changed. The figure above shows that the execution time of the new function is 0.161s which is 32 times faster than the previous function.



```
[header 1]
Parameter1
Parameter2
...
[header 2]
Units Parameter1
Units Parameter2
...
[header 3]
Coment
[header 4]
Information Parameter1
Information Parameter2
...
[header 5]
Column name [Parameter1 Parameter2 ...]
[header 6]
Value11 Value21 ...
Value12 Value22 ...
```

Figure 27: Example of a vbo file (size reduced)

To understand the difference, the figure above shows the content of a vbo file. There are headers (in yellow) with information below and the lines after the last header (in red) contained the data.

On the original function the file is read line by line to find the header and each line is stored. After this, the data is parses and finally converted from string into numeric format that can be read by Matlab.

The new function *vboread.m* reads the file line by line and directly stores the information into a structure of cells. Cells are used because the dynamic memory allocation takes less time and less than a classical array. Once the header [*data*] is found, it is assumed that the rest of the file will be only a numeric table. This allow the function to read all the data in one operation with the *dlmread* function which is used for reading databases. This function will automatically convert all the data into a numeric table.
The struct of cells is finally converted into a new struct with the same output as the previous reading function ***vboload_file [5]***.
With this new function, the whole set of analysis time is now 5 times faster.

6.2.2.2 ref_correction

This analysis step was previously done with Microsoft Excel and now implemented in Matlab. However, for a better clarity, the explanation of the function will be done with Excel tables.

R		C	R			R
TIRE1	TIRE2	TIRE3	TIRE4	TIRE5	TIRE6	TIRE7
28.32	25.78	27.46	30.41	26.92	27.16	28.86
28.82	26.63	27.68	28.77	27.34	27.14	29.26
29.51	27.31	27.79	28.73	27.60	27.35	28.65
29.06	26.85	28.41	28.71	27.70	27.01	27.66
28.97	27.15	28.03	28.45	27.77	27.93	28.79
29.10	27.19	28.20	29.35	27.54	27.58	28.55
29.35	27.09	28.46	29.75	27.32	27.75	28.69
29.33	26.80	29.15	29.25	27.95	27.92	28.88
	26.89	28.44	28.47	28.00	27.83	28.65
	26.80			27.89	27.65	28.90
					27.96	

Figure 28: Example of a data set (Excel file)

The function takes as input several data points measured for different tires. There are reference tires (R: Column 1,4,7 in this case (yellow)) and a control tire (C: Column 3 (green))

First, the average of each column is calculated.

29.1	**26.9**	**28.2**	**29.1**	**27.6**	**27.6**	**28.7**

Figure 29: Average values

Then, a linear interpolation of the average of each reference tire is calculated.

Note: (Interpolated values in the figures are rounded).

29.1	29.1	29.1	29.1	29.0	28.8	28.7

Figure 30: Interpolated reference values

Next, each average value (figure29) is divided by the corresponding interpolated reference value (figure30).

100.0%	108.3%	103.2%	100.0%	104.9%	104.6%	100.0%

Figure 31: Rating vs reference

Then, the average values (figure31) are divided by the average value of the control tire (green figure31).

96.9%	104.9%	100.0%	96.9%	101.7%	101.3%	96.9%

Figure 32: Rating vs control

Finally, all the individual values are normalized (figure 33) by the ratio of the rating vs the control (figure 32).

96.9%	104.9%	100.0%	96.9%	101.7%	101.3%	96.9%
R		C	R			R
99.4%	109.3%	102.6%	92.7%	104.2%	102.8%	96.3%
97.7%	105.8%	101.8%	98.0%	102.6%	102.9%	95.0%
95.4%	103.1%	101.4%	98.1%	101.7%	102.1%	97.0%
96.9%	104.9%	99.2%	98.2%	101.3%	103.4%	100.5%
97.2%	103.7%	100.5%	99.1%	101.1%	100.0%	96.6%
96.7%	103.6%	99.9%	96.1%	101.9%	101.3%	97.4%
95.9%	104.0%	99.0%	94.8%	102.7%	100.6%	96.9%
96.0%	105.1%	96.7%	96.4%	100.4%	100.0%	96.3%
	104.7%	99.1%	99.0%	100.2%	100.4%	97.0%
	105.1%			100.6%	101.0%	96.2%
					99.9%	

Figure 33: Normalized data

6.2.2.3 *remove_wrong_data*

This function was previously done manually. It is implemented to remove dropouts or peaks in a set of braking data (Figure 34). It is done by using smoothing and fixed tolerances.

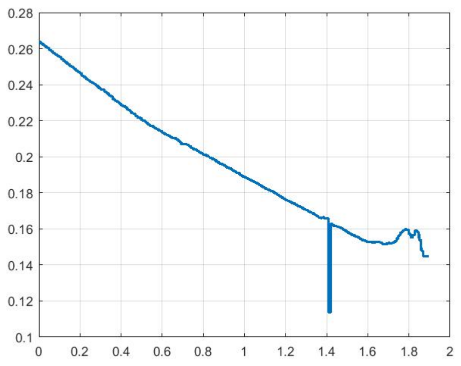
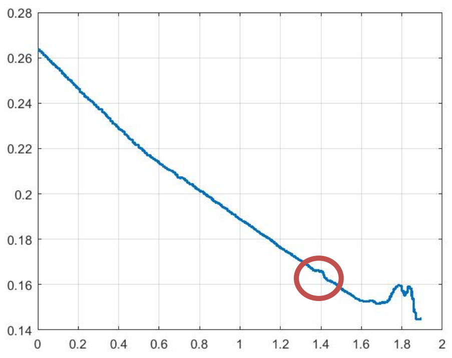



Figure 34: Example of a dropouts in a data set before and after correction

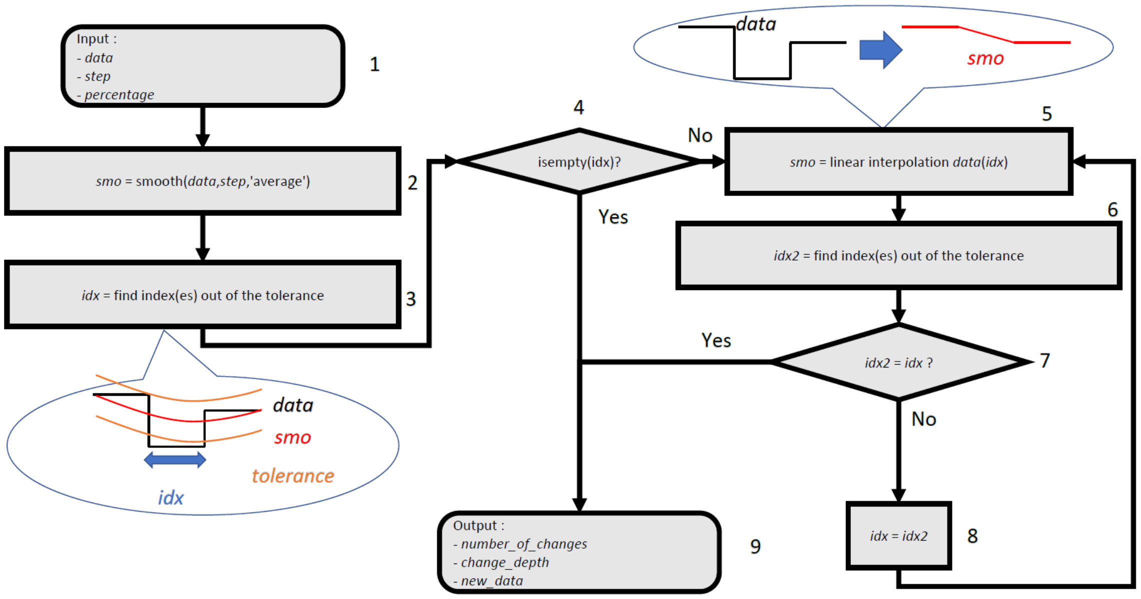


Figure 35: Diagram of the function

The figure 35 shows how the function works. First, the user need to set the allowed percentage of variation (*percentage*) between two consecutive points in the data (*data*). Then the user need to set the variable *step* corresponding to the maximum size of a dropout.

The function will do a first approximation (*smo*) to find the indexes (*idx*) out of the tolerance (block2). If no dropouts are found, the function will return the data without changing anything (block9). If dropouts are found, it will apply a linear interpolation between the last valid point and the next valid point following the potential dropout (block5). The linear interpolation loop (block 5 - 8) is performed to prevent removal of valid points near dropouts.

The new data with the linear interpolation (*smo*) will then be compared to the raw data (*data*) (block6). If the same indexes of dropouts are found twice (block7), the function will return the number of changes (*number_of_changes)*, the indexes of the modified points (*change_depth)*, and the corrected data (*new_data*) (block9). Otherwise, since the number of indexes is reduced at each iteration, the loop from block 5 to block 8 will be executed until finding the final dropout indexes.

6.2.2.4 tukey_kramer

The *Tukey's HSD (honestly significant difference)* test, was previously done with the statistical software SAS JMP.

Tukey's test compares the means of every treatment to the means of every other treatment.

The usage of this tool assumes the following:

- The observations being tested are independent within and among the groups
- The groups associated with each mean in the test are normally distributed.
- There is equal within-group variance across the groups associated with each mean in the test.

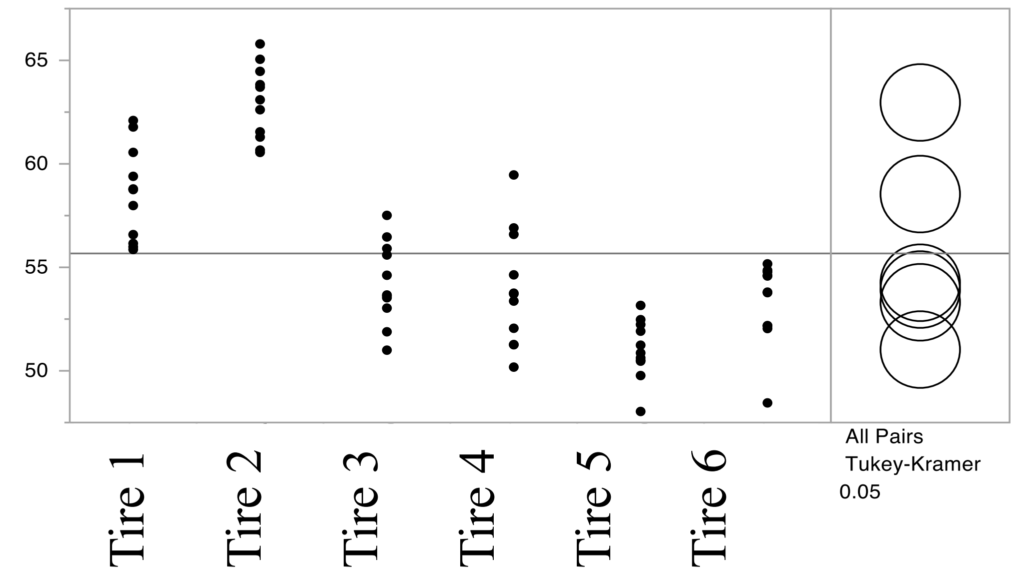


Figure 36: Example of data to test (screenshot from SAS JMP)

In figure 36 the data sets correspond to different tires which need to be compared. The software will compare each tire and group the based on the *Tukey Kramer* test result. To visualize the grouping, a *connecting letters report* is created (figure 37).

Level							Mean
Tire 2	A						62.965945
Tire 1		B					58.540364
Tire 3			C				54.256600
Tire 4			C				53.927609
Tire 6			C	D			53.311109
Tire 5				D			51.027082

Figure 37: Example of connecting letters report

On the connecting letters report, there are different groups (A, B, C, D in this example), they are sorted by descending order of the mean. This is very useful to compare different tires.

The corresponding Matlab code built during this project is *tukey_kramer.m*. The ***statistical toolbox [5]*** was necessary to build this function. The result is provided by the *mulcompare* function in numeric table. To convert this numeric table into the connecting letters report, the function *phgroup* from ***Cleiton [1]*** was modified and used.

6.3 - Validation

6.3.1 - Interface

The newly developed interface was used during two months by four different engineers. Based on the feedback of the users, improvements were made and the software is now fully operational.

6.3.2 – Code implementation

Each implemented function was tested for at least one month. Several modifications were made base on the feedback.

7 – Conclusion

The aim of the project was to group several Matlab scripts related to ABS braking analysis into one user interface. By using the possibility to create an interface and compile it with Matlab, all scripts can now be launched from the GUI. Furthermore, the users needed to have information available on how to interpret the figures. For this specific purpose a function was developed.

Improvement were made by adding or modifying different functions. based on feedback from the users. All functions covered by external software are now implemented in Matlab. In addition, the execution time has been significantly decreased.

From a personal perspective, this project was a complete success, each point in the original project plan was completed and even more. Organizing meetings, presenting and explaining my work to experienced engineers was very rewarding, particularly when all feedback was positive. The international aspect of the company allowed me to improve my communication skills.

8 - References

[1] Cleiton: https://stackoverflow.com/questions/12717635/how-to-perform-a-tukeyhsd-test-in-matlab-and-get-the-grouped-pairs-using-letters

[2] Deploytool: http://nl.mathworks.com/help/compiler/deploytool.html

[3] Geodetic toolbox: https://nl.mathworks.com/matlabcentral/fileexchange/15285-geodetic-toolbox?focused=6513461&tab=function

[4] mlapp2classdef: https://nl.mathworks.com/matlabcentral/fileexchange/56237-mlapp2classdef

[5] Statistical toolbox: https://fr.mathworks.com/help/stats/

[6] vboload_file: https://www.vboxautomotive.co.uk/index.php/en/customer-area/software#data-analysis-and-post-processing

9 - Glossary

ABS:	Antilock Braking System
EMEA:	Europe Middle East Africa
GIC-L:	Goodyear Innovation Center Luxembourg
GUI:	Graphical User Interface
LCM:	Life Cycle Mechanics

10 – Annex

Codes are in alphabetic order. (Will be added at the end of the internship)

ABBA.m

```
classdef ABBA < matlab.apps.AppBase
  properties (Access = public)
    UIFigure                matlab.ui.Figure
    TabGroup                matlab.ui.container.TabGroup
    CheckBox                matlab.ui.control.CheckBox
    Tab                     matlab.ui.container.Tab
    LabelEditField          matlab.ui.control.Label
    EditField               matlab.ui.control.EditField
    Button                  matlab.ui.control.Button
    Buttonhelp              matlab.ui.control.StateButton
    Button2                 matlab.ui.control.Button
    CloseFigureButton       matlab.ui.control.Button
    TextArea                matlab.ui.control.TextArea
    Picture                 matlab.ui.control.Button
    end
  % Store ini file into struct
  properties (Access = private)
    screen1 = ini_to_struct('Script_1.ini');
    screen2 = ini_to_struct('Script_2.ini');
    screen3 = ini_to_struct('Script_3.ini');
    screen4 = ini_to_struct('Script_4.ini');
    screen5 = ini_to_struct('Script_5.ini');
    screen6 = ini_to_struct('Script_6.ini');
  end
  % function corresponding to Button or interface component
  methods (Access = private)
    % Code that executes after component creation
    function startupFcn(app)
      value = 0;
      display('ABBA is running');
    end
    % Code that executes after pushing help button
    function helpfunc(app)
      if app.TabGroup.SelectedTab.Children(2).Value==1;
        app.TabGroup.SelectedTab.Children(1).Visible = 'on';
      else
        app.TabGroup.SelectedTab.Children(1).Visible = 'off';
      end
    end
    % Code that executes after pushing start button
    function startfunc(app)
      value = str2num(app.TabGroup.SelectedTab.Tag);
      app.savefunc()
      switch value
        case 1
          Script_1;
          Dhfclass('Script_1', 'Pictures/figurehelp.mat');
        case 2
          Script_2;
          Dhfclass('Script_2', 'Pictures/figurehelp.mat');
        case 3
          Script_3;
          Dhfclass('Script_3', 'Pictures/figurehelp.mat');
        case 4
          Script_4;
          Dhfclass('Script_4', 'Pictures/figurehelp.mat');
        case 5
          Script_5;
          Dhfclass('Script_5', 'Pictures/figurehelp.mat');
        case 6
          Script_6;
          Dhfclass('Script_6', 'Pictures/figurehelp.mat');
      end
    end
    % Code that executes after pushing save button
    function savefunc(app)
      %choose struct corresponding to the selected Tab
      value = str2num(app.TabGroup.SelectedTab.Tag);
      switch value
        case 1
          s = app.screen1;
        case 2
          s = app.screen2;
        case 3
          s = app.screen3;
        case 4
          s = app.screen4;
        case 5
          s = app.screen5;
         case 6
          s = app.screen6;
      end
        %store Text value and Hidepropertie into a struct
      for i=3:3:length(app.TabGroup.SelectedTab.Children(1:(end-2)))
        s.(char(app.TabGroup.SelectedTab.Children(i+1).Text)).Value =
app.TabGroup.SelectedTab.Children(i).Value;
        s.(char(app.TabGroup.SelectedTab.Children(i+1).Text)).Text =
app.TabGroup.SelectedTab.Children(i+2).Text;
        s.(char(app.TabGroup.SelectedTab.Children(i+1).Text)).Hide =
app.TabGroup.SelectedTab.Children(i+2).Value;
      end
      %write into the ini file
      struct_to_ini(app.TabGroup.SelectedTab.Title,s);
    end
    function CloseFigureButtonfunc(app)
      close all
    end
    function webfunc(app)
    web('www.website.com');
    end

    end
  % App Construction
  methods (Access = private)
    % Create UIFigure and components
    function createComponents(app)
```

```
        NET.addAssembly('System.Windows.Forms');
        rect = System.Windows.Forms.Screen.PrimaryScreen.Bounds;
        screensize2 = [rect.Width rect.Height];
        Window_size1 = 8;
        Window_size2 = 50;
        Window_size3 = screensize2(1)-11;
        Window_size4 = screensize2(2)-78;
        Tab_size3 = screensize2(1)-15;
        Tab_size4 = screensize2(2)-180;
        check_size1 = 300;
        check_size2 = Window_size4-10;
        check_size3 = 350;
        check_size4 = 800;
        check_pos1 = check_size1-290;
        Font_size = [20];
        Start_postion = [Tab_size3/2 5];
        Close_postion = [Tab_size3*2/3 5];
        Save_position = [Tab_size3/3 5];
        Help_position = [0.1 0.1 50 30];
        Font_size =  15;
        Button_Font_size = [25];
        Button_size = [150 44];
        Text_help_position = [50 30 800 500];
        Space_between = 25;
        Font_color = [0.88 0.88 0.88];
        Button_Font_color = [0.2 0.2 0.2];
        % Create UIFigure
        app.UIFigure = uifigure;
        app.UIFigure.Position = [Window_size1 Window_size2 Window_size3
Window_size4];
        app.UIFigure.Name = 'ABBA';
        setAutoResize(app, app.UIFigure,false);
        app.UIFigure.CloseRequestFcn = createCallbackFcn(app, @delete);
        app.UIFigure.Color = [0.12 0.12 0.12];
        app.UIFigure.Resize = 'on';

        % Create stopfigure button
        app.CloseFigureButton = uibutton(app.UIFigure, 'push');
        app.CloseFigureButton.Position = [Close_postion Button_size];
        app.CloseFigureButton.ButtonPushedFcn = createCallbackFcn(app,
@CloseFigureButtonfunc);
        app.CloseFigureButton.Text = ['Close all'];
        app.CloseFigureButton.FontSize = Button_Font_size;
        app.CloseFigureButton.BackgroundColor = Button_Font_color;
        app.CloseFigureButton.FontColor = Font_color;
        % Create Button
        app.Button = uibutton(app.UIFigure, 'push');
        app.Button.Position = [Save_position Button_size];
        app.Button.ButtonPushedFcn = createCallbackFcn(app, @savefunc);
        app.Button.Text = ['Save'];
        app.Button.FontSize = Button_Font_size;
        app.Button.BackgroundColor = Button_Font_color;
        app.Button.FontColor = Font_color;
        % Create Button2
        app.Button2 = uibutton(app.UIFigure, 'push');
        app.Button2.Position = [Start_postion Button_size];
        app.Button2.ButtonPushedFcn = createCallbackFcn(app, @startfunc);
        app.Button2.Text = ['Start'];
        app.Button2.FontSize = Button_Font_size;
        app.Button2.BackgroundColor = Button_Font_color;
        app.Button2.FontColor = Font_color;
        % Create TabGroup
        app.TabGroup = uitabgroup(app.UIFigure);
        app.TabGroup.Units = 'pixels';
        app.TabGroup.Position = [0 60 Tab_size3 Tab_size4];
        app.TabGroup.TabLocation = 'left';
        storage.(strcat('field',num2str(1))) =
app.screen1;storage.(strcat('field',num2str(7))) = 'Script_1.ini';
        storage.(strcat('field',num2str(2))) =
app.screen2;storage.(strcat('field',num2str(8))) = 'Script_2.ini';
        storage.(strcat('field',num2str(3))) =
app.screen3;storage.(strcat('field',num2str(9))) = 'Script_3.ini';
        storage.(strcat('field',num2str(4))) =
app.screen4;storage.(strcat('field',num2str(10))) = 'Script_4.ini';
        storage.(strcat('field',num2str(5))) =
app.screen5;storage.(strcat('field',num2str(11))) = 'Script_5.ini';
        storage.(strcat('field',num2str(6))) =
app.screen6;storage.(strcat('field',num2str(12))) = 'Script_6.ini';
        for i=1:length(fieldnames(storage))/2
          % Create Tab
          app.Tab(i) = uitab(app.TabGroup);
          app.Tab(i).Units = 'pixels';
          app.Tab(i).Title = storage.(strcat('field',num2str(i+6)));
          app.Tab(i).Tag = num2str(i);
          app.Tab(i).BackgroundColor = [0.20 0.20 0.20];
          app.Tab(i).ForegroundColor = [0 0 0];

          fn = fieldnames(storage.(strcat('field',num2str(i))));
          j = 0;check_size2 = Tab_size4;c = 0;htxt = '';
          while j<(length(fn))
            j = j+1;
            fnj = fn{j};
            if length(fnj)>4&fnj(1:4)=='help'
              c = c+1;
              helptxt = storage.(strcat('field',num2str(i))).(fnj).Text;
              htxt{c} = helptxt;
            else
              check_size2 = check_size2-Space_between;
              % Create CheckBox
              app.CheckBox = uicheckbox(app.Tab(i));
              app.CheckBox.Text =
storage.(strcat('field',num2str(i))).(fnj).Text;
              app.CheckBox.FontColor = Font_color;
              app.CheckBox.FontSize = Font_size;
              app.CheckBox.Position = [check_size1+check_size3-110
check_size2 check_size4+340 Font_size+5];
              if storage.(strcat('field',num2str(i))).(fnj).Hide==1
                app.CheckBox.Value = 1;
                else
                app.CheckBox.Value = 0;
              end
              % Create LabelEditField
```

```
                app.LabelEditField = uilabel(app.Tab(i));
                app.LabelEditField.Text = fn(j);
                app.LabelEditField.Position = [check_pos1 check_size2 
check_size3 Font_size+5];
                app.LabelEditField.FontSize = Font_size;
                app.LabelEditField.FontColor = Font_color;
                % Create EditField
                app.EditField = uieditfield(app.Tab(i), 'text');
                app.EditField.Position = [check_pos1+170 check_size2 
check_size3 Font_size+5];
                app.EditField.Value = 
storage.(strcat('field',num2str(i))).(fnj).Value;
                app.EditField.FontSize = Font_size;
                app.EditField.BackgroundColor = Font_color;
              end
            end
            % Create Button
            app.Buttonhelp(i) = uibutton(app.Tab(i), 'state');
            app.Buttonhelp(i).Position = Help_position;
            app.Buttonhelp(i).ValueChangedFcn = createCallbackFcn(app, 
@helpfunc);
            app.Buttonhelp(i).Text = ['help'];
            app.Buttonhelp(i).FontSize = Font_size;
            app.Buttonhelp(i).BackgroundColor = [0.5 0.5 0.5];
            % Create TextArea
            app.TextArea(i) = uitextarea(app.Tab(i));
            app.TextArea(i).Position = Text_help_position;
            app.TextArea(i).Visible = 'off';
            app.TextArea(i).Value = htxt;
            app.TextArea(i).Editable = 'on';
            app.TextArea(i).BackgroundColor = 'white';
            app.TextArea(i).FontColor = 'black';
            app.TextArea(i).FontSize = Font_size;
          end
          app.Picture = uibutton(app.UIFigure, 'push');
          app.Picture.Text = [''];
          app.Picture.Position = [0 0 223 60];
          app.Picture.Icon = 'Pictures/logo.png';
          app.Picture.ButtonPushedFcn = createCallbackFcn(app,@webfunc);
        end
      end
```

```
    % initialization and destruction
    methods (Access = public)
      % Construct app
      function app = ABBA()
        % Create and configure components
        app.createComponents()
        % Execute the startup function
        % runStartupFcn(app, @startupFcn)
        app.startupFcn()
        if nargout == 0
          clear app
        end
      end
      % Code that executes before app deletion
      function delete(obj)
          close all
        % Delete UIFigure when app is deleted
        delete(obj.UIFigure)
      end
    end
  end
```

Dhfclass.m

```
classdef Dhfclass % Display help figure class

  % Parameter declaration
  properties (Access = public)
    figuretab = {}; % Tab to store figure which are displayed
    filestore; % Store the path of the .mat file related
    store; % Structure to store load data from mat file
    initval; % flag to know if the field in the mat file exist
    fn; % name of the current file to creat a field in the struct
  end
  % Methods functions related to the item
  methods (Access = public)
    % function to load the data from the matfile and create menu
    function run(dhf)
      for i=1:length(dhf.figuretab)
        d = {}; % init d
        x = {}; % init x
        windowmenu = uimenu(dhf.figuretab(i), 'Label', 'Figure help');
% new tab item
        for j=1:length(dhf.store) % select and create the type of item
from the matfile
          if (dhf.figuretab(i).Number == str2num(dhf.store(j).Figure) &
dhf.store(j).Type == 'helpdlg')% To detect the number of the figure and
the type of item
            x = {dhf.store(j).Menu, dhf.store(j).Window,
dhf.store(j).Text};
            contentmenu = uimenu('Parent', windowmenu, 'Label',
dhf.store(j).Menu, 'UserData', x, 'Callback', @dhf.helpfig);
          elseif (dhf.figuretab(i).Number ==
str2num(dhf.store(j).Figure) & dhf.store(j).Type == 'picture')% To
detect the number of the figure and the type of item
            x = {dhf.store(j).Menu, dhf.store(j).Window,
dhf.store(j).Text};
            contentmenu = uimenu('Parent', windowmenu, 'Label',
dhf.store(j).Menu, 'UserData', x, 'Callback', @dhf.pictfig);
          end
          if length(x)==3 % To avoid issue related to the size
            d(j) = x(1); % store the name of the item in d
          else
            d = {''}; % to display nothing on the remove menu if there
is no item
          end
        end
        contentmenu = uimenu('Parent', windowmenu, 'Label', 'Add
help'); % create Contentmenu Add help on the Menu Figure help
        uimenu('Parent', contentmenu, 'Label', 'Help dialog',
'Callback', @dhf.sethelpdlgWindow); % create Contentmenu help dialog on
the Content menu help Add help
        uimenu('Parent', contentmenu, 'Label', 'Picture', 'Callback',
@dhf.setPictWindow); % create Contentmenu Picture on the Content menu
Add help
        uimenu('Parent', contentmenu, 'Label', 'Remove', 'UserData', d,
'Callback', @dhf.setMenu); % create Contentmenu Remove on the Content
menu Add help & add name of the item in the Remove menu
```

```
        end
      end
      % create menu and fields if there are nothing related on the
matfile
      function init(dhf)
        for i=1:length(dhf.figuretab)
          windowmenu = uimenu(dhf.figuretab(i), 'Label', 'Figure help');
% Create the figure help menu
          contentmenu = uimenu('Parent', windowmenu, 'Label', 'Add
help');  % create Contentmenu Add help on the Menu Figure help
          uimenu('Parent', contentmenu, 'Label', 'Help dialog',
'Callback', @dhf.sethelpdlgWindow); % create Contentmenu help dialog on
the Content menu help Add help
          uimenu('Parent', contentmenu, 'Label', 'Picture', 'Callback',
@dhf.setPictWindow); % create Contentmenu Picture on the Content menu
Add help
          uimenu('Parent', contentmenu, 'Label', 'Remove'); % create
Contentmenu Remove on the Content menu
          temp = load(dhf.filestore,'tmp');
          tmp = temp.tmp;
          tmp.(char(dhf.fn)) = []; % Create an empty struct with the
field fn
          save(dhf.filestore,'tmp','-append'); % Save into mat file
        end
      end
      function setPictWindow(dhf, src, eventData)
        x = inputdlg({'Menu Name','Window Name','Picture Name'}); % get
the date from the chatbox
        num = num2str(src.Parent.Parent.Parent.Number); % get the number
of the figure which recieve a new picture
        temp = load(dhf.filestore,'tmp'); % load data from mat file
        tmp = temp.tmp;
        dhf.store = temp.tmp.(char(dhf.fn));
        tmp.(dhf.fn) = [struct('Type', 'picture', 'Figure', num, 'Menu',
x(1), 'Window', x(2), 'Text', x(3)) dhf.store];
        save(dhf.filestore,'tmp','-append'); % Save into mat file
        if isempty(src.Parent.Children(1).Callback) % Check if the remove
menu is empty
          src.Parent.Children(1).UserData = x(1);
          src.Parent.Children(1).Callback = @dhf.setMenu; % Create the
remove menu
        else
          src.Parent.Children(1).UserData =
[src.Parent.Children(1).UserData x(1)]; % Add new item to the remove
menu
        end
        uimenu('Parent', src.Parent.Parent, 'Label', x{1}, 'UserData', x,
'Callback', @dhf.pictfig); % Add new item name to the to Menu Figure
help
      end
      function sethelpdlgWindow(dhf, src, eventData)
        x = inputdlg({'Menu Name','Window Name (Let empty for set a
Document Path)','Text (Or Document Path)'}); % get the date from the
chatbox
        num = num2str(src.Parent.Parent.Parent.Number); % get the number
of the figure which recieve a new Help dialog
```

```
      temp = load(dhf.filestore,'tmp'); % load data from mat file
      tmp = temp.tmp;
      dhf.store = temp.tmp.(dhf.fn); % Extract the struct related to
the filename
      tmp.(dhf.fn) = [struct('Type', 'helpdlg', 'Figure', num, 'Menu',
x(1), 'Window', x(2), 'Text', x(3)) dhf.store];
      save(dhf.filestore,'tmp','-append'); % Save into mat file
      if isempty(src.Parent.Children(1).Callback) % Check if the remove
menu is empty
        src.Parent.Children(1).UserData = x(1);
        src.Parent.Children(1).Callback = @dhf.setMenu; % Create the
remove menu
      else
        src.Parent.Children(1).UserData =
[src.Parent.Children(1).UserData x(1)]; % Add new item to the remove
menu
      end
      uimenu('Parent', src.Parent.Parent, 'Label', x{1}, 'UserData', x,
'Callback', @dhf.helpfig); % Add new item name to the to Menu Figure
help
    end
    function setMenu(dhf, src, eventData)
      choice = dhf.choosedialog('Select a Menu that you want to
remove', 'Remove manager', src.UserData); % get the data from the
listbox
      flag = 9999; % flag init with a big number
      for i=1:length(src.Parent.Parent.Children) % read the number of
item in the menu figure help
        if (length('Add help') ==
length(src.Parent.Parent.Children(i).Label) & 'Add help' ==
src.Parent.Parent.Children(i).Label)
          flag = i; % change flag if add help is read to allow delete
on right item
        end
        if (length(choice) ==
length(src.Parent.Parent.Children(i).Label) & choice ==
src.Parent.Parent.Children(i).Label) % check if the choice correspond
to and item
          if (i>flag)
            k = i-1;
          else
            k = i;
          end
          delete(src.Parent.Parent.Children(i)); % delete on the menu
figure list
          src.UserData(end-k+1) = []; % delete on the struct
          temp = load(dhf.filestore,'tmp'); % load data from mat file
          tmp = temp.tmp;
          dhf.store = temp.tmp.(dhf.fn); % load the previous data from
the file
          dhf.store(k) = []; % remove the selected  item
          tmp.(dhf.fn) = dhf.store;
          save(dhf.filestore,'tmp','-append'); % Save into mat file
          return;
        end
      end
    end
    % Help Window
    function helpfig(dhf, src, eventData)
      if isempty(src.UserData{2})
        winopen(src.UserData{3});% display a file
      else
        helpdlg(src.UserData{3}, src.UserData{2}); % display and help
window
      end

    end
    % Window with a picture
    function pictfig(dhf, src, eventData)
      figure('Name',src.UserData{2},'NumberTitle','off');
      imshow(['Pictures/',src.UserData{3}]); % display and picture
window
    end
    % return the choice on the choosedialog box
    function choice = choosedialog(dhf, textset, titleset, data)
    d = dialog('Position',[300 300 250 150],'Name',titleset); % Display
a list of item

    txt = uicontrol('Parent',d,...
      'Style','text',...
      'Position',[20 80 210 40],...
      'String',textset);
    popup = uicontrol('Parent',d,...
      'Style','popup',...
      'Position',[75 70 100 25],...
      'String',data,...
      'Callback',@popup_callback);
    btn = uicontrol('Parent',d,...
      'Position',[89 20 70 25],...
      'String','Delete');
    uiwait(d);
       function popup_callback(popup,event)
        idx = popup.Value;
        popup_items = popup.String;
        choice = char(popup_items(idx,:));
       end
     end
  end
  % creation and initialization of parameters and content
  methods (Access = public)
    function dhf = Dhfclass(filen, filestore)
      dhf.fn = filen;
      dhf.figuretab = get(0,'Children');
      dhf.filestore = filestore;
      temp = load(filestore,'tmp');
      if (~isfield(temp.tmp,dhf.fn) | isempty(temp.tmp.(dhf.fn))) %
check if the load struct is empty
        dhf.initval = true; % flag init
        dhf.init; % Initialize the function
      else
        dhf.initval = false; % flag init
        dhf.store = temp.tmp.(dhf.fn); % load data from mat file
```

```
            dhf.run; % Run the function
          end
        end
      end
    end
```

ini_to_struct.m

```
function [Structure]= ini_to_struct(file)
  % Parses .ini file
  % Returns a structure with section names and value, comment and if
the parameter is comment as fields.
  fileID=fopen(file);
  tline=fgetl(fileID);
  i=1;
  while tline~=-1
    [Key,Text] = strtok(tline);        % key is the first group of char
before a space, text is the rest of the line
    [Val] = strtok(Text,' = % ');      % val is the value of the
parameter
    if (Key(1)=='%'&&length(Key)~=1)   % for a comment parameter
      [lul,Text] = strtok(Text,'%');
      Key=Key(2:end);
      Structure.(char(Key)).Value = Val;
      Structure.(char(Key)).Text = Text(3:end);
      Structure.(char(Key)).Hide = 1;
    elseif (Key(1)=='%'&&length(Key)==1)% for a comment line ( exemple
: % help this thing do that)
      Structure.(char(strcat('help',num2str(i)))).Text =Text;
      Structure.(char(strcat('help',num2str(i)))).Hide = 1;
      i=i+1;
    else                                % for standard parameter
      [lul,Text] = strtok(Text,'%');
      Structure.(char(Key)).Value = Val;
      Structure.(char(Key)).Text = Text(3:end);
      Structure.(char(Key)).Hide = 0;
    end
    tline=fgetl(fileID);
  end
  fclose(fileID);
end
```

struct_to_ini.m

```
function struct_to_ini(filename,Structure)
  fid = fopen(filename,'w');
  Sections = fieldnames(Structure);% returns the Sections
  j=1;
  for i=1:length(Sections)
    Section = char(Sections(i)); % convert to character
    if Structure.(Section).Hide~=1
      fprintf(fid,'%s = %s %%
%s\n',Section,Structure.(Section).Value,Structure.(Section).Text);%
output [Section]
    elseif Section(1:4)=='help'
      fprintf(fid,'%%%s
\n',Structure.(char(strcat('help',num2str(j)))).Text);
      j=j+1;
    else
      fprintf(fid,'%%%s = %s %%
%s\n',Section,Structure.(Section).Value,Structure.(Section).Text);
    end
  end
  fclose(fid); % close file
end
```

vboread.m

```
function [vbo] = vboread(a, b)
  fileID=fopen([b, a],'r');
  tline=fgets(fileID);
  idx = 3; % minimum number of header
  date = tline;
  cname = 1; % init number header
  ccell = 1; % init number of cell in the header
  nbline = 1; % init nb line on the file
  count = 0;
  while ischar(tline)
    count = count + 1;
    if length(tline) ~= 0
      if (tline(1, 1) == '[') % to find header
        if cname > 1 & cname < idx - 1% removelast line in the field
          data.(field)(end) = [''];
        end
        tline = tline(2:end-3); % remove [] and eol char
        if (length(tline) == 4 & tline == 'data') % find header data
          idx = cname;
        else
          idx = idx + 1;
        end
        data.content1{cname} = tline;
        ccell = 1;
        cname = cname + 1;
        field = strcat('content',num2str(cname));
      end
      tline = lower(fgets(fileID));
      if cname>1 & cname < idx
        data.(field){ccell} = tline(1:end-2);
      elseif (cname == idx + 1)
        field = strcat('content',num2str(cname));
          data.(field) = dlmread([b, a], '' ,count ,0); % read the data
          % data.(field) = data.(field)(:, 1:end-1);  % remove eol char
          fclose(fileID);
          vbo = storevbodata(data, date); % store the data
          return
      end
      ccell = ccell + 1;
    else
      tline = fgets(fileID);
    end
  end
end

function vbo = storevbodata(data, date)
  run vbo_standard_channels;
  num_standard_channels = 0;
  unit_num = 1;
  standard_channels;
  vbo.sections = [];
  newsec.name = 'created';
  newsec.content = {date,''}; % '(DUMMY)'
  vbo.sections = newsec;
  vbo.channels = [];
  num_channels = length(data.content2) - 1;
  numsec = length(data.content1);
  for k =  1 : numsec
    field = ['content',num2str(k+1)];
    newsec.name = char(data.content1(k));
    if (k == 1 || k == 2 || k == numsec)
      newsec.content = '(DUMMY)';
    else
      newsec.content = data.(field);
    end
    vbo.sections = [vbo.sections newsec];
  end
  for j = 1 : num_channels
    sec.name = char(data.content2(j));
    sec.units = [];
    sec.data = data.(field)(:,j);
    sec.literal_data = [];
    vbo.channels = [vbo.channels sec];
  end
  for channel_num = 1:size(vbo.channels,2)
      for std_num = 1:size(standard_channels,1)
          if (1 ==
strcmp(standard_channels(std_num,1),vbo.channels(channel_num).name))
              num_standard_channels = num_standard_channels + 1;
              vbo.channels(channel_num).units =
char(standard_channels(std_num,2));
              break;
          end
      end
  end
  for channel_num = 1+num_standard_channels:size(vbo.channels,2)
      if (unit_num > size(data.content3, 2))
          break;                                    % not enough custom
units defined
      end
      vbo.channels(channel_num).units = char(data.content3(unit_num));
      unit_num = unit_num + 1;
  end
  for channel_num = 1:num_channels
      if (1 == strcmp('time', vbo.channels(channel_num).name))
          vbo.channels(channel_num).literal_data =
vbo.channels(channel_num).data;
          for cell = 1:size(vbo.channels(channel_num).literal_data)
              % convert string HHMMSS.ss to seconds
              literal = vbo.channels(channel_num).literal_data(cell);
              sec = mod(literal,100);
              min = mod(literal - sec, 10000) / 100;
              hour = (literal - (min*100) - sec) / 10000;
              vbo.channels(channel_num).data(cell) = hour * 3600 + min
* 60 + sec;
          end
      end
  end
end
```

ref_correction.m

```
function [normalized_data, average, interpolated_ref_distance,
rating_vs_ref, rating_vs_control, avg_check] =
ref_correction(input,iref,ictrl)

  if nargin<3
    disp('the function need 3 parameters.')
  end
  ref = find(iref == 1);
  ctrl = find(ictrl == 1);
  if isempty(ref)
    ref(1) = 1;
  end
  if isempty(ctrl)
    ctrl = ref(1);
  end
  maxv = 0;
  if iscell(input)
    for i = 1 : length(input)
      sizel(i) = length(input{i});
      maxv = max(maxv, length(input{i}));
    end
    for i = 1 : length(input)
      if (length(input{i}) < maxv)
        input{i} = [input{i};NaN(maxv-length(input{i}),1)];
      end

    end
    tab = cell2mat(input');
  end
  % init parameters
  average = [];
  interpolated_ref_distance = [];
  rating_vs_ref = [];
  rating_vs_control = [];
  avg_check = [];
  tab
  sz = size(tab);
  normalized_data = NaN(sz);
  corrected_output = NaN(sz);
  nbr = sz(1); % number of row
  nbc = sz(2);% number of column
  for i = 1:nbc
    average(i) = nanmean(tab(1:nbr,i));
    if ~isempty(intersect(ref,i))
      interpolated_ref_distance(i) = average(i);
    end
  end
  if length(ref) == 1
    interpolated_ref_distance(1:nbc) = average(ref);
  else
    if length(unique(average)) ~= 1
      for i = 2 : length(ref)
          step=(average(ref(i))-average(ref(i-1)));
          if step==0
            interpolated_ref_distance(ref(i-1):ref(i))=average(ref(i-
1));
          else
            interpolated_ref_distance(ref(i-1):ref(i)) = average(ref(i-
1)):(average(ref(i))-average(ref(i-1)))/(ref(i)-ref(i-
1)):average(ref(i));
          end
      end
    else
        interpolated_ref_distance = average;
    end
  end
  for i = 1:nbc
    % rating_vs_ref(i) = interpolated_ref_distance(i)/average(i);
    if interpolated_ref_distance(i) ~= 0
        rating_vs_ref(i) = average(i)/interpolated_ref_distance(i);
    else
        rating_vs_ref(i) = inf;
    end
  end
  for i = 1:nbc
    if rating_vs_ref(ctrl) ~= 0
      rating_vs_control(i) = rating_vs_ref(i)/rating_vs_ref(ctrl);
    else
      rating_vs_control(i) = inf;
    end
  end
  for i = 1:nbc
    if interpolated_ref_distance(i) ~= 0
        normalized_data(:,i) =
tab(:,i)/rating_vs_ref(ctrl)./interpolated_ref_distance(i);
    else
        normalized_data(:,i) = inf;
    end
  end
  for i = 1:nbc
    avg_check(i) = nanmean(normalized_data(:,i));
  end
  tmp = normalized_data;
  normalized_data = {};
  for i = 1:nbc
    tp = tmp(:, i);
    if sizel(i) ~= maxv
      tp(sizel(i) + 1 : end) = [];
    end
    if unique(tab(1:sizel(i),i) == 0);
      tp(1:sizel(i)) = 1;
    end
    normalized_data{i, 1} = tp;
  end
end
```

remove_wrong_data.m

```
function [number_of_change, change_depth, z] = remove_wrong_data(x, 
step, y) % y percent error tolerance <1

% step should be greater than the biggest set of wrong data
  % initialization
  smo = smooth(x,step,'moving'); % smoothed data
  z = x;
  change_depth = [];
  idx = [];
  id = [];
  number_of_change = 0;
  idx = find(abs(x-smo)>mean(x)*(y)); % find indices where the data is 
out of the tolerance smoothed data y/2*min
  if isempty(idx)
    return;
  end
  idx(end+1) = 0;
  id = find((diff(idx)==1) == 0);
  number_of_change = length(id); % number of  0 = number  of wrong set 
of point
  init = 1;
  change_depth(1:2,1)=[0,0];
  for i = 1 : number_of_change
    tmp = x(idx(id(i))+1) - x(idx(init)-1); % index of the table out of 
the tolerance for the i set of wrong data
    if tmp == 0
      tmp = 1;
    end
    change_depth(1:2,i) = [idx(init) idx(id(i))];
    z(idx(init)-1:idx(id(i))+1) = (x(idx(init)-1):tmp/(idx(id(i))-
idx(init)+2):x(idx(id(i))+1)); % take the linear adverage value of good 
data
    init = id(i) + 1;
  end
  number_of_change=1;
    idx2 = [];
  for c = 1:99

    smo = z;
    idx = find(abs(x-smo)>mean(x)*(y));% find indices where the data is 
out of the tolerance y
    if size(idx) == size(idx2)
      if idx == idx2
        return;
      end
    end
    change_depth = [];
    z = x;
    idx2 = idx;
    idx(end+1) = 0;
    id = find((diff(idx)==1) == 0);
    number_of_change = length(id); % number of  0 = number  of wrong 
set of point
```

```
    init = 1;
    change_depth(1:2,1)=[0,0];
    for i = 1 : number_of_change
      tmp = x(idx(id(i))+1) - x(idx(init)-1);
      if tmp == 0
        tmp = 1;
      end
      change_depth(1:2,i) = [idx(init) idx(id(i))];
      z(idx(init)-1:idx(id(i))+1) = (x(idx(init)-1):tmp/(idx(id(i))-
idx(init)+2):x(idx(id(i))+1)); % take the linear adverage value of good
data
      init = id(i) + 1;
    end
  end

end
```

tukey_kramer.m

```
function [Letters_Report, Differences_Report] = tukey_kramer(name,
data, alpha)
  if iscell(name)
    name = char(name);
  end
  if iscell(data)
    data = cell2mat(data);
  end

  [~,tbl,stats] = anova1(data,name ,'off');

  [c,m,h,gnames] = multcompare(stats,'Alpha',alpha,'CType','hsd');

  [resh order] = sort(m(:,1) + 2.061 * m(:,2),'descend');

  stats.gnames = stats.gnames(order);

  stats.n = stats.n(order);

  stats.means = stats.means(order);

  [c,m,h,gnames] = multcompare(stats,'Alpha',alpha,'CType','hsd');

  dr = [stats.gnames(c(:,1)) stats.gnames(c(:,2)) num2cell(c(:,4))
num2cell(c(:,3)) num2cell(c(:,5))  num2cell(c(:,6))];

  [resh2 order2] = (sort(abs(cell2mat(dr(:,3))),'descend'));

  dr = dr(order2,:);
  idx = find(cell2mat(dr(:,3))<0);
  if ~isempty(idx)
    tmp = dr(idx, 1);
    dr(idx, 1) = dr(idx, 2);
    dr(idx, 2) = tmp;
    dr(idx, 3:5) = num2cell(-cell2mat(dr(idx, 3:5)));
    tmp = dr(idx, 4);
    dr(idx, 4) = dr(idx, 5);
    dr(idx, 5) = tmp;
  end

  % Group data with letters
  TG = phgroup(c);

  Letters_Report = [gnames num2cell(m(:,1)) TG];
  [resh order] = sort(m(:,1),'descend');
  Letters_Report = Letters_Report(order,:);

  Differences_Report = dr;
end
% Function from https://stackoverflow.com/questions/12717635/how-to-
perform-a-tukeyhsd-test-in-matlab-and-get-the-grouped-pairs-using-
letters
function TG = phgroup(c)
```

```
%
% c: matrix of pairwise comparison results from multcompare test
% Getting significant pairwise comparisons
gr=1;
for i=1:size(c,1)
  if c(i,3)>0&&c(i,5)>0||c(i,3)<0&&c(i,5)<0

    tt(c(i,2),c(i,1))=0;
    gr=gr+1;
  else
    tt(c(i,1),c(i,1))=gr;
    tt(c(i,2),c(i,1))=gr;
  end
end
%
% Setting groups if all non-significant
if isempty(find(tt>0))==1
  for i=1:size(tt,1)
      gr=gr+1;
      tt(i,i)=gr;
  end
end

% Setting groups if some non-significant
for i=1:size(tt,1)
    if isempty(find(tt(i,:)>0))==1
        tt(i,i)=gr+1;
        gr=gr+1;
    end
end
% Correcting repeated groups
for i=1:size(tt,2)-1
    if max(find(tt(:,i+1)>0))==max(find(tt(:,i)>0))
        tt(find(tt(:,i+1)>0),i+1)=tt(i,i);
    end
end

mx=max(tt);
for i=1:size(tt,2)-1
    if max(tt(:,i+1))==mx(i)
        tt(find(tt(:,i+1)>0),i+1)=0;
    end
end
% Setting sequential groups
[B,IX] = sort(nonzeros(max(tt))');
for l=1:size(tt,1)
    for c=1:size(tt,2)
        if tt(l,c)>0
            for u=1:size(B,2)
                if tt(l,c)==B(u)
                    tt(l,c)=IX(u);
                end
            end
        end
    end
end
```

```
  % Assigning letters to groups
gn=['A';'B';'C';'D';'E';'F';'G';'H';'I';'J';'K';'L';'M';'N';'O';'P';'Q'
;'R';'S';'T';'U';'V';'W';'X';'Y';'Z'];
  for i=1 : size(tt,1)
  tg=[];
      ttu = nonzeros(unique(tt(i,:)))';
      for j= 1 : size(ttu,2)
              tg = [tg gn(ttu(1,j))];
              TG{i,1} = tg;
      end
  end
end
```